Non-Interactive Proofs for Integer Multiplication

Ivan Damgård and Rune Thorbek

BRICS, Dept. of Computer Science, University of Aarhus

Abstract. We present two universally composable and practical protocols by which a dealer can, verifiably and non-interactively, secret-share an integer among a set of players. Moreover, at small extra cost and using a distributed verifier proof, it can be shown in zero-knowledge that three shared integers a, b, c satisfy $ab = c$. This implies by known reductions non-interactive zero-knowledge proofs that a shared integer is in a given interval, or that one secret integer is larger than another. Such primitives are useful, e.g., for supplying inputs to a multiparty computation protocol, such as an auction or an election. The protocols use various set-up assumptions, but do not require the random oracle model.

1 Introduction

Applications such as auctions, elections or benchmarking analysis all involve computing on confidential data from several parties who do not trust each other a priori. This means that solutions involving a single trusted party are typically unsatisfactory. In principle, all such problems can be solved using general secure multiparty computation [18, 2, 8], where all parties take part in computing the desired results. But in practice, this is often not realistic: in auctions or elections, for instance, the number of parties holding inputs can be very large, they cannot be assumed to be expert users nor can their machines be assumed to be on-line at particular times. Hence assuming that all such parties can reliably take part in a multi-round protocol is unrealistic.

It is therefore often suggested that a smaller number of servers are assigned to do the computation, acting effectively as representatives for the clients supplying inputs. Of course, this makes sense only if the complexity of supplying inputs is much smaller than the complexity of taking part in the actual computation. In particular, we would want that supplying inputs is non-interactive. This problem can be solved using a non-interactive verifiable secret sharing (VSS) scheme. Having done the VSS's, the servers hold shares of all inputs and can do the computation using any of the (numerous) known multiparty computation techniques. Several non-interactive VSS protocols are known see, e.g., [22].

However, many applications require that the inputs supplied satisfy certain constraints. These constraints are typically phrased in a natural way as relations over the integers, because the underlying application is a computation on integers. This is the case for auctions, elections and many statistical applications such as benchmarking. For instance, an auction might specify that bids have to be in a certain interval. In other types of auctions (so called double auctions[4]), a bid consists of a sequence of numbers that must be monotonely increasing.

Standard efficient techniques for handling this would have a client commit to his input and prove in zero-knowledge that his numbers satisfy the required relations. But this solution requires interaction in its basic form. The interaction can typically be removed following the Fiat-Shamir heuristic if we are willing to assume the random oracle model. However, it is well known that the security guarantee provided by a proof in the random oracle model leaves something to be desired: we cannot instantiate the oracle with a concrete function and be sure that this always works. Hence, our goal is to avoid random oracles and still have an efficient solution.

In [5], Boudot presents an efficient technique to prove relations, as outlined above, given a primitive to prove that a committed integer is a square. Furthermore, in [1], Abe, Cramer and Fehr propose efficient and non-interactive techniques for proving multiplicative relations on secret-shared values, using distributed-verifier proofs. Unfortunately, the protocols and definition from [1] are not directly useful in the scenarios outlined above, for several reasons: First, the relations that can be proved only hold modulo some (public) prime number, and not necessarily over the integers. Second, for the case of honest majority, the protocols in [1] are only "non-interactive with complaints", that is, if a server is unhappy with the data he received privately from the dealer, he will complain, and the dealer must intervene in a second round to resolve these conflicts. It is clear that we have to avoid this in our scenario. Third, the definition of distributed verifier proofs used in [1] works with only one prover. In our scenario, we will have many provers, some of which may be corrupted. In contrast to the single-prover case, a corrupt prover may now try to exploit the information sent by honest provers in order to cheat.

In this paper, we propose two protocols that allow a client to non-interactively VSS integers among the servers, and prove in zero-knowledge, by a distributed verifier proof, that shared integers a, b, c satisfy $ab = c$. Using known reductions [5], this implies non-interactive proofs that a shared integer is in a given interval, or that shared numbers a, b satisfy $a \geq b$. Both protocols require one broadcast from the prover and one round of messages between the verifiers (servers), which is a minimal amount of interaction for a distributed verifier proof. Details on the communication complexity of the protocols follow below. We prove our protocols secure in the Universal Composability model (with static adversary), this automatically gives us a definition handling the multiple prover case.

For the first solution, we take the protocol of [1] as the point of departure, introducing new techniques to solve the problems mentioned above. We obtain our solution by replacing in the protocol from [1] Shamir secret-sharing by Linear Integer Secret Sharing (LISS) [13] – which exists for any access structure [13]. LISS schemes are basically secret sharing schemes where the secret is reconstructed by taking a integral linear combination of the shares. Also, we replace Pedersen commitments [22] by the integer commitments from [15].

While this is quite straightforward, it is not so trivial to solve the problem of handling complaints without interaction. We first observe that the reason why the dealer must resolve conflicts in the protocol by Abe et al. is that only point-to-point channels between dealer and each server are assumed, and hence servers are not a priori committed to what they received. On the other hand, a typical implementation would realize the channels using public-key encryption, so we propose to include this encryption explicitly in the protocol. One might now hope that a server can prove it received bad data by "opening" the ciphertexts it received. However, while the sender of a ciphertext can always "open" it convincingly (simply by revealing the coins used to create it), we need that the *receiver* can do so. Since ciphertexts can be adversarially generated, and unopened ciphertexts must remain secure, it is not immediately clear how this can be done in a non-interactive and efficient way. We propose an efficient solution to the problem based on Identity-Based Encryption (IBE). To our knowledge, this is a new application of IBE, and we believe the idea is of independent interest, as the possibility of "complaining convincingly" is often useful in protocol design.

For the case of honest majority, the VSS we obtain requires the dealer to send a total of $O(n \log n(\kappa + l + k + n))$ bits, where κ is the security parameter for the public-key and commitment schemes used, n is the number of players, l is the bit length of the numbers we share and k is an "information theoretic" security parameter, controlling the statistical leakage of information.

The protocol can handle any Q2 adversary structure (honest majority in the threshold case), which is optimal in terms of the number of corruptions that can be handled at all. However, for realistic values of the parameters, the efficiency is not what we might hope for. This is because the numbers we will be computing on will be numbers specifying bids, prices, productions costs, etc., that is, numbers that are typically much smaller than those used for public-key cryptography. Realistic parameter values might be $n = 7$, $l = 32$, $k = 60$ and $\kappa = 1024$. In such a case, each 32 bit number we share is expanded to about 25.000 bits, which hardly seems desirable.

We therefore propose another solution, where we make the stronger assumption that the adversary structure is Q3 (less than $n/3$ corruptions in the threshold case). We build a solution using a generalization of the pseudorandom secret-sharing technique from [10] to the case of linear *integer* secret sharing. In the threshold case, the protocol requires the dealer to send, once and for all, $O(T(\kappa + nk))$ bits to the servers, where T is the number of maximal unqualified sets in the adversary structure. After this, any number of VSS's can be done by sending $O(l + k)$ bits to the servers for each value to be shared. Each multiplication proof requires 3 VSS invocations and in addition $O((l + k + n)n)$ bits should be sent.

The initial step is not always efficient as a function of n because T may be exponential in n, depending on the adversary structure. In the typical threshold case, T would be about $\binom{n}{n/3}$. But for a small number of servers, T is moderate. On the other hand, for fixed n and for a large number of VSS invocations we come very close to sending only $l + k$ bits for every l-bit number we share - where of course sending l bits is necessary. It is therefore ideally suited for cases, where a large number of clients need to supply large amounts of data to a small number of servers. For the example parameter values above and assuming we share, say 200 numbers, the dealer needs to send about 230 bits per number to share.

Both our protocols use a common reference string, and assume that the verifiers have public/secret key pairs set up in advance. Note that if we do not assume random oracles, we cannot get non-interactive protocols without some sort of set-up assumption. Of course, using set-up assumptions, our problem could also be solved using standard techniques for non-interactive zero-knowledge. But with current state of the art, this approach can only prove the type of statements we are after using generic techniques. This would give non-interactive proofs of size $\Omega(l\kappa|C|)$ where $|C|$ is the size of a Boolean circuit C checking the relation in question. For realistic parameter values, this will be several orders of magnitude larger than our complexity. To our knowledge, our solutions are the first non-interactive protocols for integer relations that do not use random oracles, and have communication complexity independent of the circuit complexity of the relation.

2 Preliminaries

In a Linear Integer Secret Sharing (LISS) Scheme there are n players, which are denoted by $P_1, \ldots, P_n$. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of all the players, and let the power set of $\mathcal{P}$ be denoted by $P(\mathcal{P})$. Let $s \in [-2^l..2^l]$ be the secret which a dealer D wants to secret share between the players in $\mathcal{P}$ over a LISS. Then the sets in $P(\mathcal{P})$ which are allowed to reconstruct the secret s are called *qualified* and the sets which should not be able to obtain any information about the secret s are called *forbidden*.

Definition 1. *The collection of qualified sets, $\Gamma \subseteq P(\mathcal{P})$, is called a* monotone access structure*, if for all $A \in \Gamma$ and $A \subset B \subseteq \mathcal{P}$ it holds that $B \in \Gamma$.*

We also need the notion of an *adversary structure* [19].

Definition 2. *An* adversary structure *is a monotone collection of sets, $\Delta \subseteq P(\mathcal{P})$, for which the adversary may corrupt the players of one set in the adversary structure. It is monotone in the sense that for every $A \in \Delta$ it holds that for every $B \subset A$ that $B \in \Delta$.*

Definition 3. *An adversary structure Δ is $Q2$ ($Q3$) if no two (three) sets in the structure cover the full player set $\mathcal{P}$.*

If Γ is the collection of all qualified sets of players in $\mathcal{P}$ and Γ is a monotone access structure, then the corresponding adversary structure, Δ, is the collection of all the forbidden sets. Note that, Δ is monotone as required by an adversary structure, and that $\Gamma \cup \Delta = P(\mathcal{P})$ and $\Gamma \cap \Delta = \emptyset$. That is, an adversary structure can be seen as a complement of a monotone access structure. Since the structures, Γ and Δ, are monotone, they can be uniquely represented by their minimal and maximal sets denoted by Γ^- and Δ^+, respectively. $|\Delta^+|$ will denote the number of sets in Δ^+. In this paper we use Γ and Δ interchangeably. We proceed to define what is meant by a *correct* and *private* LISS.

Definition 4. *A LISS scheme is* correct*, if the secret can be reconstructed from shares of any qualified set in $A \in \Gamma$, by taking an integer linear combination of the shares with coefficient that depends only on the index set A.*

Definition 5. *A LISS scheme is* private*, if for any forbidden set $B \in \Delta$, any two secret $s, s' \in [-2^l..2^l]$, and independent random coins r and r', the statistical distance between the distributions of the shares $\{s_i(s, r, k) \mid i \in B\}$ and $\{s_i(s', r', k) \mid i \in B\}$ is negligible in the* security *parameter k.*

A *labeled matrix* consists of a $d \times e$ matrix M and a corresponding surjective function $\psi : \{1, \dots, d\} \to \{1, \dots, n\}$. We say that the i-th row is *labeled* by $\psi(i)$ or *owned* by player $P_{\psi(i)}$. For any subset $A \subset \mathcal{P}$, we let M_A denote the restriction of M to the rows labeled by some $P_{\psi(i)} \in A$. For any d-vector $\boldsymbol{x}$, we similarly denote $\boldsymbol{x}_A$ to be the restriction of entries i with $P_{\psi(i)} \in A$. For any two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, let $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ denote the inner product.

Definition 6. *An* Integer Span Program (ISP) for a monotone access structure *Γ consists of a tuple $\mathcal{M} = (M, \psi, \boldsymbol{\varepsilon})$, where $M \in Z^{d,e}$ is a labeled matrix with a surjective function $\psi : \{1, \dots, d\} \to \{1, \dots, n\}$, and the* target vector *$\boldsymbol{\varepsilon} = (1, 0, \dots, 0)^T \in Z^e$. Furthermore, for every $A \subseteq \mathcal{P}$ the following holds,*

- *for every $A \in \Gamma$ there exists a* reconstruction vector *$\boldsymbol{\lambda} \in Z^d$ such that $M_A^T \boldsymbol{\lambda} = \boldsymbol{\varepsilon}$.*
- *for every $A \notin \Gamma$ there exists a* sweeping vector *$\boldsymbol{\kappa} \in Z^e$ such that $M_A \boldsymbol{\kappa} = \boldsymbol{0}$ and $\langle \boldsymbol{\kappa}, \boldsymbol{\varepsilon} \rangle = 1$.*

The size of $\mathcal{M}$ is defined to be d.

In [13] it was shown how to construct a correct and private LISS scheme from any ISP. For a given ISP we define $l_0 = l + \lceil \log_2(\kappa_{\max}(e-1)) \rceil$, where $\kappa_{\max} = \max\{|a| \mid a$ is an entry in some sweeping vector $\}$. To share a secret $s \in [-2^l..2^l]$, we use a *distribution vector* $\boldsymbol{\rho}$ which is a uniformly random vector in $[-2^{l_0+k}..2^{l_0+k}]^e$ with the restriction that $\langle \boldsymbol{\rho}, \boldsymbol{\varepsilon} \rangle = s$. The *share vector* is computed by $M\boldsymbol{\rho} = \boldsymbol{s} = (s_1, \dots, s_d)^T$, where the *share component* s_i is given to player $P_{\psi(i)}$ for $1 \le i \le n$. The *share* of player P_j is the subset of share components $\boldsymbol{s}_{\{P_j\}}$.

See [13] for a proof of correctness and privacy. There, it was also shown that LISSs exist for any adversary structure, and in particular they can be constructed for threshold structures where a player's share is $O((l+k+n^2)\log n)$ bits long. It follows from results and conjectures in [11] that this can probably be improved to $O((l+k+n)\log n)$ bits.

3 Verifiable Secret Sharing (VSS) and Distributed Verifier Proofs

3.1 Model and Definition

We have a set of dealers $\{D_1, \ldots, D_m\}$ and a set of n players or verifiers $\mathcal{P} = \{P_1, \ldots, P_n\}$. We assume an active and static adversary who may corrupt any number of dealers and a set of players in a given adversary structure. All players, dealers and the adversary are polynomially bounded. We assume (for simplicity) synchronous communication. We use the Universal Composability framework [6] and define ideal functionalities as follows:

Functionality F_{VSS}

- On input s from D_j, send ("D_j, input") to all players and the adversary. Wait one round (this models the fact that our implementation takes one round to finish, after the prover has spoken). Then, if $s = \perp$ (which may be the case if D_j is corrupt), send ("D_j, Fail") to all players, else send ("D_j, OK") to all players.

Functionality $F_{ab=c}$

- On input a, b, c from D_j, send ("D_j, input") to all players and the adversary. Wait one round. Then, if a, b, c are integers satisfying $ab = c$, send ("D_j, OK") to all players, else send ("D_j, Fail") to all players.

Both functionalities need to model that a successfully shared secret can be reconstructed. To simulate this we add a command to the functionalities, where it will send the requested shared value to everyone if asked by all honest players.

For our protocols, we will need a set-up assumption, namely $D_1, \ldots, D_m$ and $P_1, \ldots, P_n$ get common input $k, pk, pk_1, \ldots, pk_n$, where k is the security parameter, pk_i is the public key of P_i, and pk is a common reference string. As private input, P_i has a secret key sk_i corresponding to pk_i. For simplicity, we assume here that the public and secret keys are generated and given to players initially by an ideal functionality T. But we stress that T can be implemented by a once-and-for-all preprocessing among the players (it is well known that any UC functionality can be securely implemented if we have honest majority, or in general $Q2$). In Section 3.4, it is even sufficient that players generate their own key pairs and broadcast the public keys. We also assume a functionality F_{BC}, allowing any dealer to broadcast information to the verifiers[1]. Communication between verifiers uses standard authenticated but non-secret channels. Note that the UC framework incorporates, in addition to the adversary Adv attacking the protocol, an environment Z that chooses inputs for and receives outputs from honest players. We will only consider environments that give integers (and not $\perp$) as input to honest players. This models the assumption that honest players would only attempt to VSS valid integers.

[1] Note, that even if we implement the broadcast via a subprotocol, this can be done such that we maintain the non-interactive nature of our proofs, namely the dealer sends a single (signed) message to all players, who then internally agree on what he said.

3.2 An Integer Commitment Scheme

A *commitment scheme* for domain $\mathcal{S}$ is given by a family of functions $\mathsf{com}_{pk} : \mathcal{S} \times \mathcal{R}_{pk} \to \mathcal{C}_{pk}$, indexed by a *public key pk*. One commits by publishing $C = \mathsf{com}_{pk}(s,r)$, where $s \in \mathcal{S}$ is the committed value and $r \in \mathcal{R}_{pk}$ is a random value. A *homomorphic* commitment scheme is a scheme where we assume that $\mathcal{S}$ is an additive group and that for any two commitments C and C' and any number λ, anyone can compute commitments S and P such that being able to open C and C' to s and s', respectively, allows to open S to the sum $s+s'$ and P to the product λs.

We use a modified version of the Pedersen commitment scheme [22], based on a multiplicative group G of order unknown to the players. This commitment scheme first appeared in [16] and later in [15]. We will need primes p, q where $p = 2p' + 1$ and $q = 2q' + 1$ and p', q' are also prime. The computations are done in Z_n^*, where $n = pq$, and the public key is $pk = (n, g, h)$ where g, h are chosen at random in Q_n, the set of squares modulo n. Then we use $\mathsf{com}_{pk} : (s,r) \mapsto g^s h^r \bmod n$. The scheme is *homomorphic*, since given commitments $C = \mathsf{com}_{pk}(s,r)$ and $C' = \mathsf{com}_{pk}(s',r')$ then $CC' = \mathsf{com}_{pk}(s+s', r+r')$ and $C^\lambda = \mathsf{com}_{pk}(\lambda s, \lambda r)$. Note that if we choose r uniformly random from $[0..n2^k]$, then $r \bmod \mathrm{ord}(h)$ is statistically close to being uniformly random in $[0..\mathrm{ord}(h)-1]$.

An important advantage of this scheme is that it allows commitment to *integers*. This follows since the commitment is done in a group G of unknown order. More specifically, the following proposition holds for the above commitment scheme.

Proposition 1 ([16]). $\mathsf{com}_{pk}(s,r)$ *is a statistically hiding and computationally binding commitment scheme, i.e.:*

- *If factoring is infeasible, then given* $pk = (n, g, h)$ *it is infeasible to compute* $s, s', r, r' \in Z$ *where* $s \neq s'$ *such that* $\mathsf{com}_{pk}(s,r) = \mathsf{com}_{pk}(s',r')$.
- *For any two values* s, s', *the distributions* $(pk, \mathsf{com}_{pk}(s,r)), (pk, \mathsf{com}_{pk}(s',r'))$ *are statistically indistinguishable.*

3.3 Public-key Encryption with Verifiable Opening

We introduce here a tool that we will need later. Suppose a player P has a public/secret key pair (pk, sk), and receives ciphertext from various senders, some of whom may be corrupt. We want that the cryptosystem is chosen ciphertext (CCA) secure and has the additional property that for any received ciphertext c, P can reveal the decryption result $x = D_{sk}(c)$ and prove non-interactively and efficiently that x is correct. We want, of course, that "unopened" ciphertexts remain secure, which excludes the trivial solution of revealing the secret key.

Note that if c is a valid ciphertext, the random coins used to generate c can serve as proof of what the plaintext was. But even if the receiver could compute these coins efficiently, there is still a problem if the sender is corrupt. Then c may be invalid, and "the coins used to generate c" is not even a well-defined notion.

A formal definition of the notion we are after can be phrased as a variant of the standard chosen ciphertext security game, where the oracle answers decryption queries with the result as well as the proof of correctness. We do not give it here for lack of space. Instead, we give our solution in a form tailored for direct use in our protocol below. The proof that it works is then incorporated in the proof for the overall protocol[2].

The key pair (pk, sk) for P will be the master secret and public key for an identity-based cryptosystem (IBE)[3]. Note that, under reasonable assumptions, efficient IBE's exist that do not use random oracles[24]. For the IBE we use, we need that given identity t and pk, one can easily verify if a secret key sk_t is the secret key for identity t. This can indeed be done for all known efficient IBE's, we call this IBE with verifiable secret keys (IBE-VSK). We assume that the system is used in a protocol that assigns a unique tag to each ciphertext to be sent to P. To encrypt message m, the sender treats the tag t for this ciphertext as an identity and encrypts the message to this id, i.e., he sends $c = E_t(m)$. The receiver decrypts by computing the secret key sk_t and then $m = D_{sk_t}(c)$. To reveal the result of decrypting c, P reveals sk_t. Everyone can now compute $D_{sk_t}(c)$. One must also verify that sk_t is indeed the secret key corresponding to t. From the assumption that tags are not reused and standard properties of IBE, it follows that unopened ciphertexts remain secure. A somewhat similar idea was used for a different purpose in [7].

3.4 VSS using Integer Commitments

In this section we construct a non-interactive *verifiable secret sharing* [9] (VSS) scheme based on LISS. We use the model described in the previous sections. Specifically, the common reference string will be a public key $pk = (g, h, n)$ for the integer commitment scheme described above. Moreover, each player P_j has a key pair (pk_j, sk_j) for an IBE-VSK as described above.

Protocol $\mathsf{VSS}_{pk}(s)$

On input $s \in [-2^l..2^l]$, the dealer D makes a commitment $C = \mathsf{com}_{pk}(s, r)$ to s, and then executes the following protocol to prove that he knows how to open C to value s, and to secret share s:

Protocol $\mathsf{Proof}_{g,h}(C)$

1. Given an ISP $\mathcal{M} = (M, \psi, \boldsymbol{\varepsilon})$, the dealer D chooses a random vector $\boldsymbol{\rho} \in [-2^{l_0+k}..2^{l_0+k}]^e$ with $\langle \boldsymbol{\rho}, \boldsymbol{\varepsilon} \rangle = s$, and commits to this *sharing vector* $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_e)^T$ by commitments $R_1, \ldots, R_e$ to $\rho_1, \ldots, \rho_e$, respectively, where $R_1 = C$ and all commitments use (g, h) as public parameter. The commitments $R_2, .., R_e$ to the additional randomness are included in the proof π. D computes the shares of s:

[2] The problem could also be solved using non-interactive zero-knowledge, but this will be much too inefficient for our purposes. Using OAEP might work as well, but only assuming random oracles which we want to avoid

$\boldsymbol{s} = (s_1, \ldots, s_d)^T = M\boldsymbol{\rho}$, and computes the opening information o_i for the corresponding commitment

$$C_i = \prod_{j=1}^{e} R_j^{m_{ij}}$$

using the homomorphic property, where m_{ij} is defined by $M = [m_{ij}]$. Finally, he includes $c_i = E_{pk_{\psi(i)}}(o_i)$ in his proof π, where all these ciphertexts are assigned a tag consisting of C concatenated with the name of D (see Section 3.3). Finally, D broadcasts C, π.

2. For each i, $P_{\psi(i)}$ decrypts c_i. If he finds that the resulting opening information o_i is incorrect w.r.t. C_i, then he sends o_i to all other players, along with a proof that o_i is indeed the result of decrypting c_i, this counts as an accusation against D. Otherwise he sends "accept".
3. For any accusation from $P_{\psi(i)}$, each player verifies that any o_i received is indeed the value that c_i decrypts to. If this is not the case this o_i is discarded.
4. Each player looks at all (non-discarded) o_i-values he knows. If any such o_i is inconsistent with C_i, then he rejects. Otherwise he accepts.

A successfully shared value s can be reconstructed by simply having every player P_i open every commitment C_j where $\psi(j) = i$. For some qualified set of successfully opened shares the players can then use the corresponding reconstruction vector $\boldsymbol{\lambda}$ to reconstruct the secret. We have

Theorem 1. *Given a secure IBE-VSK, the protocol* $\mathsf{VSS}_{pk}(s)$ *securely implements* F_{VSS}*, assuming any* $Q2$ *adversary structure* Γ*.*

Proof. To show that $\mathsf{VSS}_{pk}(s)$ securely implements F_{VSS}, we are given an adversary Adv and an environment Z, and we need to construct a simulator S. The simulator interacts with Adv to simulate its view of attacking the protocol, and on the other hand interacts with F_{VSS} on behalf of corrupt players. This game is called the *ideal process*. This is compared to the *real process*, where Z, Adv are interacting with a real instance of the protocol. In both processes, Z and Adv may communicate at any time. The goal is now to show that Z cannot distinguish the real from the ideal process. Our simulator works as follows:

1. The simulator generates the keys $pk, \{(pk_j, sk_j)\}$ following T's algorithm, and sends all public keys to Adv, along with secret keys for corrupted players.
2. The simulator S now acts whenever required, as follows:
   – If Adv sends C and a proof π to the broadcast functionality on behalf of corrupt dealer D_j, the simulator does the following: using its secret keys, it can decrypt ciphertext in π intended for honest players and follow their algorithm to compute what they would send in the second round. This also lets it decide if the proof would be accepted. If not, the simulator sends $\perp$ to F_{VSS}. If the proof is acceptable, observe first that since Γ

is $Q2$, the set of honest players, A, is qualified, and that every honest player can open his commitment to s_i. Let $\boldsymbol{\lambda}$ be a reconstruction vector for A, that is, $\langle \boldsymbol{s}, \boldsymbol{\lambda} \rangle = s$ and $\boldsymbol{\lambda}_{A^C} = \mathbf{0}$, i.e., if $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_d)^T$ then

$$\sum_{i=1}^{d} s_i \lambda_i = \sum_{i=1}^{d} \lambda_i \sum_{j=1}^{e} m_{ij} \rho_j = \rho_1 = s,$$

where $\lambda_j = 0$ for $\psi(j) \notin A$. Hence, the above equation implies that $\sum_{i=1}^{d} \lambda_i m_{ij} = \delta_{1j}$, where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. Therefore, by the homomorphic property, the simulator can open commitment $C' = \prod_{i=1}^{d} C_i^{\lambda_i}$ to $s' = \sum_{i=1}^{d} \lambda_i s_i$. Now, since

$$C' = \prod_{i=1}^{d} C_i^{\lambda_i} = \prod_{i=1}^{d} \left(\prod_{j=1}^{e} R_j^{m_{ij}} \right)^{\lambda_i} = \prod_{j=1}^{e} R_j^{\sum_i \lambda_i m_{ij}} = R_1 = C,$$

we see that the simulator can extract from the proof a way to open commitment C to a value s. The simulator sends s to F_{VSS}.

– On input ("D_j, input") from F_{VSS}, where D_j is honest, the simulator simulates what D_j would send in the protocol, as follows: First, create a commitment C to an arbitrary value. By the statistical hiding property, there exists a way to open C to the correct value s used by D_j, except with negligible probability – although s is unknown to S. We therefore proceed, assuming implicitly that C "contains" s. Now, let A be the set of corrupted players. Then there exists a sweeping vector $\boldsymbol{\kappa}$ such that $M_A \boldsymbol{\kappa} = \mathbf{0}$ and $\langle \boldsymbol{\kappa}, \boldsymbol{\varepsilon} \rangle = 1$. Let $\boldsymbol{\rho}_0 = (r_1, \ldots, r_e)^T$ be a random distribution vector such that $\langle \boldsymbol{\rho}_0, \boldsymbol{\varepsilon} \rangle = 0$, i.e., a distribution vector to a random sharing of 0. Construct $R'_1, \ldots, R'_e$ as random commitments of $r_1, \ldots, r_e$, respectively, with the exception that $R'_1 = 1$ (or the commitment of $r_1 = 0$ using randomness 0). Then, by the homomorphic property of the commitment scheme, compute commitments

$$C'_i = \prod_{j=1}^{e} {R'_j}^{m_{ij}},$$

to shares s_i which determines the secret 0. Now, given the commitment C for the secret s, we modify the commitments so they become consistent with s: Compute the public commitments $R_i = R'_i C^{\kappa_i}$ where $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_e)^T$ is the sweeping vector for A. Note that $R_1 = R'_1 C^{\kappa_1} = 1C^1 = C$ as required, since $\langle \boldsymbol{\kappa}, \boldsymbol{\varepsilon} \rangle = 1$ (i.e., $\kappa_1 = 1$). The commitments to the shares in s will be as follows:

$$C_i = \prod_{j=1}^{e} R_j^{m_{ij}} = \prod_{j=1}^{e} (R'_j C^{\kappa_j})^{m_{ij}} = \prod_{j=1}^{e} {R'_j}^{m_{ij}} C^{\kappa_j m_{ij}}.$$

For the players in A we have that,

$$\prod_{j=1}^{e} C^{\kappa_j m_{ij}} = C^{\sum_j \kappa_j m_{ij}} = C^0 = 1,$$

since the inner product of $\boldsymbol{\kappa}$ and a row in M which is owned by a player in A is 0. So for a corrupt $P_{\psi(i)}$ we have $C'_i = C_i$, and we know how to open these commitments. The simulated proof therefore consists of the commitments $R_1, \ldots, R_e$, encryptions of correct opening information for C_i when $P_{\psi(i)}$ is corrupt, and encryptions of random values for honest players.

To see that this simulation works, note the following: First, the simulation of the initial set-up stage and of the case where a corrupt dealer gives a proof is perfect. In particular, when a corrupt dealer does a VSS that would be accepted in the real protocol, the simulator can *always* extract the correct secret, and honest players will therefore output accept also in the ideal process.

In the case where an honest dealer does a VSS, this will in the ideal process simply mean that it sends integer s to F_{VSS}. The functionality will send accept to everyone, so all honest players output accept. This is also the case in the real protocol: correct opening information for each C_i is uniquely determined from the ciphertext c_i, hence no honest player will accuse D and every other accusation will be rejected by the honest players.

Hence the only possible difference between the ideal and real process is in the simulated commitment C and proof π that is shown to Adv. By the statistical hiding property of the commitment scheme and privacy of the LISS scheme, it follows that the opening information sent to corrupt players, as well as the commitments $R_1, \ldots, R_e$ have distribution statistically close the one seen in the real protocol. So the only difference is the fact that the ciphertexts intended for honest players are random in the simulation, and contain valid openings of commitments in the real protocol.

We cannot argue that the two sets of encryptions are indistinguishable based directly on the ideal process because S knows all secret keys. Instead, we construct a machine S' that acts as an adversary breaking the underlying IBE-VSK. S' will run the algorithms of Z, Adv and S, with the following modifications to S: S' receives public keys for the honest players from an oracle. Whenever S needs to decrypt a ciphertext sent to an honest player with tag t (see Section 3.3), S' will ask the oracle for the secret key for that tag, and can then decrypt. When S wants to create ciphertext for honest players in a simulated proof, S' will ask the oracle to encrypt either 1) random data or 2) genuine opening information for the relevant commitments. The latter is possible because S' also runs Z and therefore knows each secret that is shared, this allows it to create the commitment C as a genuine commitment containing the right value, and from this it can compute how to open all the other commitments in that VSS. In the case 1), we produce exactly what we get in the ideal process, in case 2) we produce something statistically close to what we get in the real process. Hence, if Z could distinguish the two processes, S' can use the output from Z to break the underlying IBE-VSK. $\square$

For lack of space, we do not prove formally here that the protocol for reconstruction of the committed secret works. It is quite straightforward based on the binding property of the commitment scheme.

3.5 Verifiable Commitment Multiplication Proof

We now show a (distributed verifier) proof that VSS'ed integers s, s', s'' satisfy that $s'' = ss'$:

Protocol $\mathsf{MultProof}_{pk}(s, s', s'')$

1. The prover makes commitments C, C', C'' to s, s', s'' and then executes $\mathsf{Proof}_{g,h}(C)$, $\mathsf{Proof}_{g,h}(C')$, and $\mathsf{Proof}_{g,h}(C'')$.
2. The prover executes $\mathsf{Proof}_{C',h}(C'')$ using the *same* distribution vector $\boldsymbol{\rho}_s$ as in step 1 (but with new independent randomness for the commitments).
3. Every player verifies whether his shares obtained from $\mathsf{Proof}_{g,h}(C)$ (from step 1.) and $\mathsf{Proof}_{C',h}(C'')$ (from step 2.) coincide. If this does not hold, he accuses the dealer by opening the ciphertexts he received in $\mathsf{Proof}_{g,h}(C)$ and $\mathsf{Proof}_{C',h}(C'')$. Each player verifies any accusation made.
4. The proof is accepted if all subproofs were accepted, and no valid accusations were made.

Note that the four executions of the Proof protocol can be run in parallel. A similar protocol appeared in [1], but we have here added $\mathsf{Proof}_{g,h}(C')$[3].

Theorem 2. *Assuming the integer commitment scheme is binding and given a secure IBE-VSK,* $\mathsf{MultProof}_{pk}(s, s', s'')$ *securely implements* $F_{ab=c}$ *assuming any* $Q2$ *adversary structure* Γ.

Proof. Note that making commitments C, C', C'' and then executing the first 3 instances of Proof is equivalent to executing 3 instances of VSS_{pk}. Therefore, to simulate this, we run the simulator from the previous theorem 3 times (in parallel). To simulate the execution of $\mathsf{Proof}_{C',h}(C'')$, we run the same simulator again, with the following changes: when simulating the actions of an honest dealer, the simulator will not create its own commitment to play the role of the commitment to the secret, instead it will use C''. Also, it will use the same distribution vector that was used in the simulation of $\mathsf{Proof}_{g,h}(C)$.

To show that this simulation works, we only need to check that when we extract opening information from an acceptable proof given by a corrupt prover, we will get values s, s', s'' such that $ss' = s''$. Note, that if the proof is accepted, it follows from the proof of Theorem 1 that we can extract from step 1. pairs $(s, r), (s', r')$ and (s'', r'') such that $C = \mathsf{com}_{g,h}(s, r)$, $C' = \mathsf{com}_{g,h}(s', r')$ and $C'' = \mathsf{com}_{g,h}(s'', r'')$. Furthermore, steps 2. and 3. ensure that we can extract (s, r^*) such that $C'' = \mathsf{com}_{C',h}(s, r^*) = C'^s h^{r^*}$ [4]. Combining this with the expression for $C' = \mathsf{com}_{g,h}(s', r') = g^{s'} h^{r'}$ we get $C'' = C'^s h^{r^*} = (g^{s'} h^{r'})^s h^{r^*} = g^{ss'} h^{r's+r^*}$ In other words, we can now open C'' to both s'' and ss', which contradicts the binding property unless $s'' = ss'$. □

[3] This is necessary since the order of the group of the commitments is unknown and we can therefore not prove soundness the same way as in [1] (Lemma 1).

[4] Note that the proof in step 2. uses C', which might have been adversarially generated, in place of g which comes from the common reference string. However, this is not a problem since the extraction will work for any set of values.

4 Verifiable Multiplication Proof Based on Pseudo-Random Sharing

4.1 Replicated Integer Secret-Sharing and Share Conversion

In this section we first introduce RISS, an integer version of Replicated Secret-Sharing [20], where we share an integer over a monotone access structure. Then we define share conversion, and show that shares generated by a RISS scheme can be locally converted to shares in the same secret generated by LISS schemes.

Scheme Replicated Integer Secret-Sharing (RISS)

Let Δ be an adversary structure. For each set $B \in \Delta^+$ choose a uniformly random r_B integer from the interval $[-2^{l+k}..2^{l+k}]$ and send privately r_B to each player $P_i \notin B$. Furthermore, publish $r = s + \sum_{B \in \Delta^+} r_B$, where s is the secret from the interval $[-2^l..2^l]$.

Lemma 1. *The RISS scheme is correct and (statistically) private.*

Definition 7. *Let $\mathcal{S}$ and $\mathcal{S}'$ be two secret-sharing schemes. We say that $\mathcal{S}$ is* locally convertible *to $\mathcal{S}'$ if there exist local conversion functions $g_1, \dots, g_n$ such that the following holds. If $(s_1, \dots, s_n)$ are valid shares of a secret s in $\mathcal{S}$, then $(g_1(s_1), \dots, g_n(s_n))$ are valid shares of the same secret s in $\mathcal{S}'$. We denote by g the concatenation of all g_i, namely $g(s_1, \dots, s_n) = (g_1(s_1), \dots, g_n(s_n))$, and refer to g as a share conversion function.*

Note by the locality feature of the conversion, that converted shares cannot reveal more information about s than the original shares.

The following theorem is proved in the full version of the paper [14], using ideas similar to what was used in [10]

Theorem 3. *The RISS scheme $\mathcal{R}_\Gamma$, realizing Γ, is locally convertible to any LISS realizing an access structure $\Gamma' \subseteq \Gamma$.*

Clearly, for any prime p, a RISS sharing of integer s can be thought of as a replicated sharing over Z_p of $s \bmod p$, by reducing all shares modulo p. Furthermore, in [10] it was shown how to locally convert a replicated sharing over Z_p to any linear secret sharing (LSS) scheme over Z_p (such as Shamir's scheme). From these two observations, we immediately get

Proposition 2. *The RISS scheme $\mathcal{R}_\Gamma$, realizing Γ, is locally convertible to any LSS over Z_p realizing an access structure $\Gamma' \subseteq \Gamma$, where the original secret s after conversion will be $s \bmod p$.*

4.2 Application to VSS

We now show how the results from the previous subsection can be used to generate a series of verifiably shared secrets by broadcasting only two values per secret, at the initial cost of distributing a set of random seeds to the players. We use the model defined earlier, where each player P_i has a public and a secret key. In this case, we assume that there is a public key pk_B defined for each $B \in \Delta^+$, and P_i's public key consists of all pk_B for those B in which P_i is *not* a member. The secret key consists of all secret keys corresponding to relevant pk_B's. As before, we assume these are keys for an IBE-VSK.

The following protocol does the intial distribution of seeds.

Protocol $\mathsf{Random}_{\{r_B\}}(\Delta^+)$

1. For each $B \in \Delta^+$ the dealer D choose an uniformly random r_B from $[0..2^k[$.
2. For each $B \in \Delta^+$ D broadcasts r_B encrypted under pk_B. The dealer's name is used as tag for this ciphertext. Each player decrypts all the ciphertexts for which he has the secret key.

The protocol clearly ensures that players have mutually consistent shares, i.e., all honest players not in B agree on the value of r_B, for any $B \in \Delta^+$.

Given a pseudorandom function (PRF) $\varphi_{\cdot}(\cdot)$ with k-bit keys and inputs, and outputs in $[-2^{l+k}..2^{l+k}]$, the following protocol is realizable.

Protocol $\mathsf{VSS}_{\{r_B\}}(s)$

It is assumed that the dealer D has run $\mathsf{Random}_{\{r_B\}}(\Delta^+)$ on some adversary structure, Δ.

1. D broadcasts a value a, to serve as a "label" for this instance of the protocol. The only demand is that a can be used as input to φ, and that D never reuses an a-value. D computes, with his knowledge of $\{r_B\}$, $r = s + \sum_B \varphi_{r_B}(a)$ and broadcasts r.
2. Each player P_i checks that $r \in [-(|\Delta^+| + 1)2^{l+k}..(|\Delta^+| + 1)2^{l+k}]$, and rejects if this is not the case. Otherwise, he computes $\varphi_{r_B}(a)$, for every B where $P_i \notin B$.

This lemma follows easily by inspection of the protocol:

Lemma 2. *If D is honest, no honest player will reject in $\mathsf{VSS}_{\{r_B\}}(s)$. No matter what the dealer does, if honest players accept, the set of values $r, \{\varphi_{r_B}(a)|\ B \in \Delta^+\}$ form a RISS sharing of some value s'. If D is honest, $s' = s$, otherwise $s' \in [-(2|\Delta^+| + 2)2^{l+k}..(2|\Delta^+| + 2)2^{l+k}]$.*

It is also quite straightforward to see that if D is honest, and the PRF is secure, a polynomially bounded adversary does not learn anything about the secret involved. A proof of this is implicit in the proof of Theorem 4 below. We discuss in the full version of this paper [14] how a secret can be reconstructed, once it has been VSS'ed as above.

4.3 Multiplication Proof

In this section we describe a protocol which non-interactively proves that a shared value is the product of two other shared values. For simplicity, we will only consider the case of a threshold adversary who corrupts $t < n/3$ of the players, so the adversary structure Δ will in this section consist of all set of cardinality at most t. The full version of this paper [14] will describe a generalization to all Q3 adversary structures.

We will need a tool from [10], called Pseudorandom Zero Sharing (PRZS). This protocol assumes that for all $B \in \Delta^+$, players not in B have been given t random seeds $r_B^1, \ldots, r_B^t$ and a prime $p > n$ is agreed in advance. Based on this, the protocol generates (by local computation only) a pseudorandom polynomial f over Z_p of degree at most $2t$ such that $f(0) = 0$ and each player P_i knows $f(i)$. The protocol is a simple generalization of the share conversion technique.

In the following $\mathsf{Random}_{\{r_B, r_B^1, \ldots, r_B^t\}}(\Delta^+)$ will denote the protocol where the dealer distributes the seeds $r_B, r_B^1, \ldots, r_B^t$ to all players not in B using encryption under pk_B. We will choose a fixed prime p, such $p > 2(4|\Delta^+| + 2)^2 2^{2(l+k)}$.

Protocol $\mathsf{MultProof}_{\{r_B, r_B^1, \ldots, r_B^t\}}(a, b, c)$

1. The dealer D executes $\mathsf{Random}_{\{r_B, r_B^1, \ldots, r_B^t\}}(\Delta^+)$.
2. D executes $\mathsf{VSS}_{\{r_B\}}(a)$, $\mathsf{VSS}_{\{r_B\}}(b)$ and $\mathsf{VSS}_{\{r_B\}}(c)$.
3. The players use Proposition 2 to locally convert the RISS sharings we now have of a, b, c to Shamir sharings of $a \bmod p$,$b \bmod p$ and $c \bmod p$, consistent with polynomials f_a, f_b and f_c of degree at most t, t and $2t$ respectively. The players use PRZS to generate shares in a polynomial f of degree at most $2t$ with $f(0) = 0$.
4. D uses his knowledge of all seeds to compute the polynomial $h = f + f_a f_b - f_c$ and broadcasts h.
5. Each player P_i verifies that $h(i) = f(i) + f_a(i) f_b(i) - f_c(i)$. If the verification fails then P_i broadcast "Accusation" and opens all encrypted values $r_B, r_B^1, .., r_B^t$ known by him.
6. The proof is rejected if one of the following situations happen: one of the VSS protocols in Step 2 was rejected, the broadcasted polynomial h is not of degree at most $2t$, $h(0) \neq 0$, or broadcasted values by a player are consistent with the encrypted values but inconsistent with the broadcasted values by D.

Theorem 4. *When based on a secure IBE-VSK and PRF, then the protocol* $\mathsf{MultProof}_{\{r_B, r_B^1, \ldots, r_B^t\}}(a, b, c)$ *securely implements* $F_{ab=c}$*, for any threshold-t adversary structure where* $t < n/3$.

Proof. We construct a simulator S that works as follows:

1. S generates the keys $pk, \{(pk_B, sk_B)\}$ following T's algorithm, and sends all public keys to Adv, along with secret keys for corrupted players.
2. S now acts whenever required, as follows:
   - When Adv does a proof on behalf of a corrupt dealer, S can simply decrypt everything sent by the adversary, and decide if the proof would be accepted in the real process. If so, it reconstructs values a, b and c and sends them to the ideal functionality. Otherwise, it sends $\perp$ to the ideal functionality and uses the honest players' algorithm to compute

the messages (complaints) they would send to corrupt players, and sends these to Adv.
- When an honest dealer does a proof, S will generate a simulated proof by simply following the prover's algorithm, using $a = b = c = 0$.

To see that this simulation works as required, note first that the simulation of the set-up phase and proofs by corrupt dealers is perfect. This is because the simulator follows the honest players algorithm to compute their reaction to the proof, so we just need to check that when the proof is accepted, the simulator can send a correct witness to the functionality. By Lemma 2, the values a, b, c that the simulator reconstructs from the proof will be in the interval $[-(2|\Delta^+| + 2)2^{l+k} \ldots (2|\Delta^+| + 2)2^{l+k}]$, so we know that $|ab|, |c|$ are less than $p/2$. Now, from Step 5, we know that h agrees with $f + f_a f_b - f_c$ in all points owned by honest players, of which there are at least $2t + 1$. This implies that $h = f + f_a f_b - f_c$, and therefore that $ab = c \bmod p$. But if $ab \neq c$, it would have to be the case that $|ab - c| \geq p$, while on the other hand we already know that $|ab - c| \leq |ab| + |c| < p$. So indeed $ab = c$.

It remains to show that the simulation of an honest dealer's proof shown to the adversary is indistinguishable from a real proof. For this, consider the real process $Real$, and assume the worst case where the adversary has corrupted a maximal set B of players. This means that when an honest dealer does a proof, the key sk_B is the only secret key the adversary does not know. We then define a new "hybrid" process Hyb_1, where we replace the broadcasted encryptions of $r_B, r_B^1, \ldots, r_B^t$ (under pk_B) by encryptions of independent random values. By an argument similar to the proof of Theorem 1, $Real$ is indistinguishable from Hyb_1 if the underlying IBE-VSK is secure. Note that in Hyb_1, we can replace evaluations of the PRF using seeds $r_B, r_B^1, \ldots, r_B^t$ by oracle access to the PRF with the same seeds, and all messages sent will remain unchanged. We define Hyb_2 by replacing the PRF oracles by oracles for truly random functions. By security of the PRF, Hyb_2 is indistinguishable from Hyb_1. Finally, we define Hyb_3 as follows: we first replace the dealer's inputs (a, b, c) to the $\mathsf{VSS}_{\{r_B\}}(\cdot)$-protocols by random values in the legal interval, and second, we choose the polynomial h to broadcast as a uniformly random polynomial, subject to $h(0) = 0$, $deg(h) \leq 2t$, and that $h(i)$ agrees with the adversary's information for all corrupt players P_i. Now, Hyb_3 is statistically indistinguishable from Hyb_2: consider, for instance, the execution of $\mathsf{VSS}_{\{r_B\}}(a)$ in Hyb_2. If we subtract the randomness that the adversary already knows, we see that he can compute $R + a$, where R is a truly random value in $I_r = [-2^{l+k}..2^{l+k}]$. This is statistically indistinguishable from $R + r$ where r is a random value in $I_s = [-2^l..2^l]$, which is what the adversary would see in Hyb_3. The polynomial h is easily seen to have exactly the same distribution in Hyb_2 and Hyb_3. It follows that $Real$ is indistinguishable from Hyb_3.

To finish the proof, note that in the argument we just gave, we did not use anything special about the inputs a, b, c, other than $ab = c$. Therefore, essentially the same argument shows that the ideal process is also indistinguishable from

Hyb_3 since the simulator uses $a = b = c = 0$ and otherwise follows the protocol. The theorem now follows from transitivity of indistinguishability. □

5 Interval Proofs and Application to Secure Computing

Boudot [5] observes that to prove that a number x lies in an interval $[a, b]$ it is sufficient to prove that $x - a \geq 0$ and $b - x \geq 0$. By using a homomorphic commitments scheme and a primitive to prove that a committed integer is a square, he constructs an efficient proof that a committed number is non-negative. Only a small constant number of calls to the primitive is required.

Boudot's protocols can be run in our settings by using one of the VSS protocols we have presented to play the role of commitments in Boudot's protocols. Note that both types of VSS's we construct are linear and so we have the homomorphic properties needed. In this way, we get a non-interactive proof that a shared number is in a given interval, using a constant number of invocations of our VSS protocol.

Furthermore, each number x we prove something about is verifiably shared among the players, using a LISS scheme (a RISS scheme in case of the second protocol). If we consider the shares as numbers mod q for any prime q, we obtain a linear sharing over Z_q of x mod q. We can now, possibly after local conversion using [10], do secure computing on such numbers using, e.g., the protocols from [17, 4, 12]. If what we really want is secure addition and multiplication over the integers, we can use the initial interval proofs to make sure the numbers are small enough to avoid modular reductions.

Acknowledgements

We thank Matthias Fitzi, Jørgen Brandt, Mikkel Krøigård, Martin Geisler, and the anonymous referees for insightful comments.

References

1. Masayuki Abe, Ronald Cramer and Serge Fehr. *Non-interactive Distributed-Verifier Proofs and Proving Relations among Commitments.* ASIACRYPT 2002, LNCS 2501. Springer 2002.
2. M. Ben-Or, S. Goldwasser, A. Wigderson: *Completeness theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, Proc. ACM STOC '88, pp. 1–10.
3. Dan Boneh, Matthew K. Franklin: *Identity-Based Encryption from the Weil Pairing.* SIAM J. Comput. 32(3): 586-615 (2003).
4. Peter Bogetoft, Ivan Damgård, Thomas Jakobsen, Kurt Nielsen, Jakob Pagter and Tomas Toft: *A Practical Implementation of Secure Auctions based on Multiparty Integer Computation.* Proc. of Financial Cryptography 2006, Springer Verlag LNCS.
5. F. Boudot. *Efficient Proofs that a Committed Number Lies in an Interval.* EUROCRYPT'00, LNCS 1807, pp 431-444, 2000.

6. Ran Canetti: *Universally Composable Security: A New Paradigm for Cryptographic Protocols*. Proc. of FOCS 2001: pp.136-145. See also updated version on the Eprint archive, www.iacr.org.
7. Ran Canetti, Shai Halevi, and Jonathan Katz. *Chosen-Ciphertext Security from Identity-Based Encryption.* Advances in Cryptology - EUROCRYPT 2004, LNCS 3027, pp 207-222, 2004.
8. D. Chaum, C. Crépeau, I. Damgård: *Multi-Party Unconditionally Secure Protocols*, Proc. of ACM STOC '88, pp. 11–19.
9. Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. *Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (extended abstract).* In 26th Annual Symposium on Foundations of Computer Science. IEEE, 1985.
10. Ronald Cramer, Ivan Damgård, Yuval Ishai: *Share Conversion, Pseudorandom Secret-Sharing and Applications to Secure Computation.* Proc. of TCC 2005, pp. 342-362, Springer Verlag LNCS
11. Ronald Cramer, Serge Fehr and Martijn Stam: *Black-Box Secret Sharing from Primitve Sets in Algebraic Number Fields*, Proc. of Crypto 05, Springer Verlag LNCS.
12. Ivan Damgrd, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, Tomas Toft: *Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation.* Proc. of TCC 2006, pp. 285-304, Springer Verlag LNCS.
13. Ivan Damgård and Rune Thorbek. *Linear Integer Secret-Sharing and Distributed Exponentiation.* PKC'06, LNCS 3958, pp 75-90 (2006).
14. Ivan Damgård and Rune Thorbek. *Non-Interactive Proofs for Integer Multiplication* (full version), the Eprint archive, www.iacr.org (eprint.iacr.org/2007/086).
15. Eiichiro Fujisaki and Tatsuaki Okamoto. *A Practical and Provably Secure Scheme for Publicly Verifiable Secret Sharing and Its Applications.* EUROCRYPT'98, LNCS 1403,pp 32-46, 1998.
16. Eiichiro Fujisaki and Tatsuaki Okamoto. *Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations.* CRYPTO'97, LNCS 1294,pp 16-30, 1997.
17. R. Gennaro, M. Rabin, T. Rabin, *Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography*, Proc of ACM PODC'98.
18. O. Goldreich, S. Micali and A. Wigderson: *How to Play Any Mental Game or a Completeness Theorem for Protocols with Honest Majority*, Proc. of ACM STOC '87, pp. 218–229.
19. Martin Hirt and Ueli Maurer *Player Simulation and General Adversary Structures in Perfect Multiparty Computation.* Journal of Cryptology: the journal of the International Association for Cryptologic Research, volume 13, pages 31-60 (2000).
20. M. Ito, A. Saito, and T. Nishizeki. *Secret sharing schemes realizing general access structures.* Proc. IEEE Global Telecommunication Conf., Globecom 87: 99-102 (1987).
21. M. Karchmer and A. Wigderson. *On Span Programs.* In Proc. of 8th IEEE Structure in Complexity Theory, pages 102-111, 1993.
22. Torben P. Pedersen. *Non-interactive and Information-theoretic Secure Verifiable Secret Sharing.* In Advances in Cryptology - CRYPTO '91, volume 576 of Lecture Notes in Computer Science. Springer, 1991.
23. Adi Shamir. *How to share a secret.* Communication of the Association for Computing Machinery, 22(11), 1979.
24. Brent Waters: *Efficient Identity-Based Encryption Without Random Oracles.* Proc. of Eurocrypt 2005: pp.114-127, Springer Verlag LNCS.