Compact VSS and Efficient Homomorphic UC Commitments*

Ivan Damgård, Bernardo David, Irene Giacomelli, and Jesper Buus Nielsen

Dept. of Computer Science, Aarhus University

Abstract. We present a new compact verifiable secret sharing scheme, based on this we present the first construction of a homomorphic UC commitment scheme that requires only cheap symmetric cryptography, except for a small number of seed OTs. To commit to a k-bit string, the amortized communication cost is $O(k)$ bits. Assuming a sufficiently efficient pseudorandom generator, the computational complexity is $O(k)$ for the verifier and $O(k^{1+\epsilon})$ for the committer (where $\epsilon < 1$ is a constant). In an alternative variant of the construction, all complexities are $O(k \cdot polylog(k))$. Our commitment scheme extends to vectors over any finite field and is additively homomorphic. By sending one extra message, the prover can allow the verifier to also check multiplicative relations on committed strings, as well as verifying that committed vectors $\boldsymbol{a}, \boldsymbol{b}$ satisfy $\boldsymbol{a} = \varphi(\boldsymbol{b})$ for a linear function φ. These properties allow us to non-interactively implement any one-sided functionality where only one party has input (this includes UC secure zero-knowledge proofs of knowledge). We also present a perfectly secure implementation of any multiparty functionality, based directly on our VSS. The communication required is proportional to a circuit implementing the functionality, up to a logarithmic factor. For a large natural class of circuits the overhead is even constant. We also improve earlier results by Ranellucci *et al.* on the amount of correlated randomness required for string commitments with individual opening of bits.

1 Introduction

A commitment scheme is perhaps the most basic primitive in cryptographic protocol theory, but is nevertheless very powerful and important both in theory and practice. Intuitively, a commitment scheme is a digital equivalent of a secure

* The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation and also from the CFEM research center (supported by the Danish Strategic Research Council), within which part of this work was performed. Partially supported by Danish Council for Independent Research via DFF Starting Grant 10-081612. Partially supported by the European Research Commission Starting Grant 279447.

box: it allows a prover P to commit to a secret s by putting it into a locked box and give it to a verifier V. Since the box is locked, V does not learn s at commitment time, we say the commitment is *hiding*. Nevertheless, P can later choose to give V the key to the box to let V learn s. Since P gave away the box, he cannot change his mind about s after commitment time, we say the commitment is *binding*.

Commitment schemes with stand-alone security (i.e., they only have the binding and hiding properties) can be constructed from any one-way function, and already this most basic form of commitments implies zero-knowledge proofs for all NP languages. Commitments with stand-alone security can be very efficient as they can be constructed from cheap symmetric cryptography such as pseudorandom generators [Nao91].

However, in many cases one would like a commitment scheme that composes well with other primitives, so that it can be used as a secure module that will work no matter which context it is used in. The strongest form of security we can ask for here is UC security [Can01]. UC commitments cannot be constructed without set-up assumptions such as a common reference string [CF01]. On the other hand, a construction of UC commitment in such models implies public-key cryptography [DG03] and even multiparty computation [CLOS02] (but see [DNO10] for a construction based only on 1-way functions, under a stronger set-up assumption).

With this in mind, it is not surprising that constructions of UC commitments are significantly less efficient than stand-alone secure commitments. The most efficient UC commitment schemes known so far are based on the DDH assumption and requires several exponentiations in a large group [Lin11,BCPV13]. This means that the computational complexity for committing to k-bit strings is typically $\Omega(k^3)$.

Our Contribution. We first observe that even if we cannot build practical UC commitments without using public-key technology, we might still confine the use of it to a small once-and-for-all set-up phase. This is exactly what we achieve: given initial access to a small number of oblivious transfers, we show a UC secure commitment scheme where the only computation required is pseudorandom bit generation and a few elementary operations in a finite field. The number of oblivious transfers we need does not depend on the number of commitments we make later. The main observation we make is that we can reach our goal by combining the oblivious transfers with a "sufficiently compact" Verifiable Secret Sharing Scheme (VSS) that we then construct. The VSS has applications on its own as we detail below.

To commit to a k-bit string, the amortized communication cost is $O(k)$ bits. The computational complexity is $O(k)$ for the verifier and $O(k^{1+\epsilon})$ for the committer (where $\epsilon < 1$ is a constant). This assumes a pseudorandom generator with linear overhead per generated bit.[1] In an alternative variant of the construction, all complexities are $O(k \cdot polylog(k))$. After the set-up phase is done, the prover can commit by sending a single string. Our construction extends to commitment to strings over any finite field and is additively homomorphic, meaning that given commitments to strings $\boldsymbol{a}, \boldsymbol{b}$, the verifier can on his own compute a commitment to $\boldsymbol{a} + \boldsymbol{b}$, and the prover can open it while revealing nothing beyond $\boldsymbol{a} + \boldsymbol{b}$. Moreover, if the prover sends one extra string, the verifier can also check that committed vectors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ satisfy $\boldsymbol{c} = \boldsymbol{a} * \boldsymbol{b}$, the component-wise product. Finally, again by sending one extra string and allowing one extra opening, the verifier can compute a commitment to $\varphi(\boldsymbol{a})$, given the commitment to $\boldsymbol{a}$, for any linear function φ. These extra strings have the same size as a commitment, up to a constant factor.

On the technical side, we take the work from [FJN$^+$13] as our point of departure. As part of their protocol for secure 2-party computation, they construct an imperfect scheme (which is not binding for all commitments). While this is good enough for their application, we show how to combine their scheme with an efficient VSS that is compact in the sense that it allows to share several values from the underlying field, while shares only consist of a single field element. This is also known as packed secret sharing [FY92].

Our construction generalises the VSS from [CDM00] to the case of packed secret sharing. We obtain a VSS where the communication needed is only a constant factor larger than the size of the secret. Privacy for a VSS usually just says that the secret remains unknown to an unqualified subset of players until the entire secret is reconstructed. We show an extended form of privacy that may be of independent interest: the secret in our VSS is a set of ℓ vectors $\boldsymbol{s}_1, ..., \boldsymbol{s}_\ell$, each of length ℓ. We show that any linear combination of $\boldsymbol{s}_1, ..., \boldsymbol{s}_\ell$ can be (verifiably) opened and players will learn nothing beyond that linear combination. We also build two new VSS protocols, both of which are non-trivial extensions. The first allows the dealer to generate several sharings of the vector 0^ℓ. For an honest dealer, the shares distributed are random even given the extra verification information an adversary would see during the VSS. This turns out to be crucial in achieving secure multiplication of secret-shared or committed values. The second new protocol allows us to share two sets of vectors $\boldsymbol{s}_1, ..., \boldsymbol{s}_\ell$ and $\tilde{\boldsymbol{s}}_1, ..., \tilde{\boldsymbol{s}}_\ell$ such that it can be verified that $\varphi(\boldsymbol{s}_1) = \tilde{\boldsymbol{s}}_1, \ldots, \varphi(\boldsymbol{s}_\ell) = \tilde{\boldsymbol{s}}_\ell$ for a linear function φ. In the commitment scheme, this is what allows us to verify that two shared or committed vectors satisfy a similar linear relation.

Before we discuss applications, a note on an alternative way to view our commitment scheme: A VSS is essentially a multiparty commitment scheme.

[1] This seems a very plausible assumption as a number of different sufficient conditions for such PRG's are known. In [IKOS08] it is observed that such PRGs follow Alekhnovich's variant of the Learning Parity with Noise assumption. Applebaum [App13] shows that such PRGs can be obtained from the assumption that a natural variant of Goldreich's candidate for a one-way function in NC0 is indeed one-way. The improved HILL-style result of Vadhan and Zheng [VZ12] implies that such PRGs can be obtained from any exponentially strong OWF that can be computed by a linear-size circuit.

Therefore, given our observation that VSS and OT gives us efficient UC commitment, it is natural to ask whether our construction could be obtained using "MPC-in-the-head" techniques. Specifically, the IPS compiler [IPS08] is a general tool that transforms a multiparty protocol into a 2-party protocol implementing the same functionality in the OT hybrid model. Indeed, applying IPS to our VSS does result in a UC commitment protocol. However, while this protocol is somewhat similar to ours, it is more complicated and less efficient.

Applications. One easily derived application of our commitment scheme is an implementation of any two-party functionality where only one party has input, we call this a *one-sided* functionality. This obviously includes UC secure zero-knowledge proofs of knowledge for any NP relation. Our implementation is based on a Boolean circuit C computing the desired output.

We will focus on circuits that are not too "oddly shaped". Concretely, we assume that every layer of the circuit is $\Omega(\ell)$ gates wide, except perhaps for a constant number of layers. Here one may think of ℓ as a statistical security parameter, as well as the number of bits one of our commitments contains. Second, we want that the number of bits that are output from layer i in the circuit and used in layer j is either 0 or $\Omega(\ell)$ for all $i < j$. We call such circuits *well-formed.* In a nutshell, well-formed circuit are those that allow a modest amount of parallelization, namely a RAM program computing the circuit can always execute $\Omega(\ell)$ bit operations in parallel and when storing bits for later use or retrieving, it can always address $\Omega(\ell)$ bits at a time. In practice, since we can treat ℓ as a statistical security parameter, its value can be quite small(e.g., 80), in particular very small compared to the circuit size, and hence a requirement that the circuit be well-formed seems rather modest. Using the parallelisation technique from [DIK10], we can evaluate a well-formed circuit using only parallel operations on ℓ-bit blocks, and a small number of different permutations of bits inside blocks. This comes at the cost of a log-factor overhead.

Some circuits satisfy an even nicer condition: if we split the bits coming into a layer of C into ℓ-bit blocks, then each such block can be computed as a linear function of blocks from previous layers, where the function is determined by the routing of wires in the circuit. Such a function is called a block function. If each block function depends only on a constant number of previous blocks and if each distinct block function occurs at least ℓ times, then C is called *regular* (we can allow that a constant number of block functions do not satisfy the condition). For instance, block ciphers and hash functions do not spread the bits around much in one round, but repeat the same operations over many rounds and hence tend to have regular circuits. Also many circuits for arithmetic problems have a simple repetitive structure and are therefore regular.

Theorem 1. *For any one-sided two-party functionality F that can be computed by Boolean circuit C, there exists a UC secure non-interactive implementation of F in the OT hybrid model. Assuming C is well-formed and that there exists a linear overhead PRG, the communication as well as the receiver's computation is in $O(\log(|C|)|C|)$. If C is regular, the complexities are $O(|C|)$.*

We stress that the protocol we build works for any circuit, it will just be less efficient if C is not well-formed.[2] We can also apply our VSS directly to implement multiparty computation in the model where there are clients who have inputs and get outputs and servers who help doing the computation.

Theorem 2. *For any functionality F, there exists a UC perfectly secure implementation of F in the client/server model assuming at most a constant fraction of the servers and all but one of the clients may be corrupted. If C is well-formed, the* total communication complexity *is in $O(\log(|C|)|C|)$. If C is regular, the complexity is $O(|C|)$.*

We are not aware of any other approach that would allow us to get perfect security and "constant rate" for regular circuits.[3]

A final application comes from the fact that our commitment protocol can be interpreted as an unconditionally secure protocol in the model where correlated randomness is given. In this model, it was shown in [RTWW11] that any unconditionally secure protocol that allows commitment to N bits where each bit can be *individually* opened, must use $\Omega(Nk)$ bits of correlated randomness, where k the security parameter. They also show a positive result that partially circumvents this lower bound by considering a functionality $F_{com}^{N,r}$ that allows commitment to N bits where only $r < N$ bits can be selectively and individually opened. When r is $O(1)$, they implement this functionality at constant rate, i.e., the protocol requires only $O(1)$ bits of correlated randomness per bit committed to. We can improve this as follows:

Theorem 3. *There exists a constant-rate statistically secure implementation of $F_{com}^{N,r}$ in the correlated randomness model, where $r \in O(N^{1-\epsilon})$ for any $\epsilon > 0$.*

We find it quite surprising that r can be "almost" N, and still the lower bound for individual opening does not apply. What the actual cut-off point is remains an intriguing open question.

Related Work. In [DIK$^+$08], a VSS was constructed that is also based on packed secret sharing (using Shamir as the underlying scheme). This construction relies crucially on hyper invertible matrices which requires the field to grow with the number of players. Our construction works for any field, including $\mathbb{F}_2$. This would not be so important if we only wanted to commit and reveal bits: we could use [DIK$^+$08] with an extension field, pack more bits into a field element

[2] It is possible to use MPC-in-the-head techniques to prove results that have some (but not all) of the properties of Theorem 1. Essentially one applies the IPS compiler to a multiparty protocol, either a variant of [DI06] (described in [IKOS09]), or the protocol from [DIK10]. In the first case, the verifier's computation will be asymptotically larger than in our protocol, in the second case, one cannot obtain the result for regular circuits since [DIK10] has at least logarithmic overhead for any circuit since it cannot be based on fields of constant size.

[3] Using [DIK10] would give at least logarithmic overhead for any circuit, using variants of [DI06] would at best give statistical security.

and still get constant communication overhead, but we want to do (Boolean) operations on committed bits, and then "bit-packing" will not work. It therefore seems necessary to construct a more compact VSS in order to get our results. In [BBDK00], techniques for computing functions of shared secrets using both broadcast channels and private interactive evaluation are introduced. However, their constructions are based specifically on Shamir's LSSS and do not allow verification of share validity.

In recent independent work [GIKW14], Garay *et al.* also construct UC commitments using OT, VSS and pseudorandom generators as the main ingredients. While the basic approach is closely related to ours, the concrete constructions are somewhat different, leading to incomparable results. In [GIKW14] optimal rate is achieved, as well as a negative result on extension of UC commitments. On the other hand, we focus more on computational complexity and achieve homomorphic properties as well as non-interactive verification of linear relations inside committed vectors.[4]

2 Preliminaries

In this section we introduce the basic definitions and notation that will be used throughout the paper. We denote sampling a value r from a distribution $\mathcal{D}$ as $r \leftarrow \mathcal{D}$. We say that a function ϵ is negligible if there exists a constant c such that $\epsilon(n) < \frac{1}{p(n)}$ for every polynomial p and $n > c$. Two sequences $X = \{X_\kappa\}_{\kappa \in \mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa \in \mathbb{N}}$ of random variables are said to be *computationally indistinguishable*, denoted by $X \overset{c}{\approx} Y$, if for every non-uniform probabilistic polynomial-time (PPT) distinguisher D there exists a negligible function $\epsilon(\cdot)$ such that for every $\kappa \in \mathbb{N}$, $|\ Pr[D(X_\kappa) = 1] - Pr[D(Y_\kappa) = 1]\ | < \epsilon(\kappa)$. Similarly two sequences X and Y of random variables are said to be *statistically indistinguishable*, denoted by $X \overset{s}{\approx} Y$, if the same relation holds for unbounded non-uniform distinguishers.

2.1 Universal Composability

The results presented in this paper are proven secure in the Universal Composability (UC) framework introduced by Canetti in [Can01]. We consider security against static adversaries, *i.e.* all corruptions take place before the execution of the protocol. We consider active adversaries who may deviate from the protocol in any arbitrary way. It is known that UC commitments cannot be obtained in the plain model [CF01]. In order to overcome this impossibility, our protocol is proven secure in the $\mathcal{F}_{OT}$-hybrid model in, where all parties are assumed to have access to an ideal 1-out-of-2 OT functionality. In fact, our protocol is constructed in the $\mathcal{F}_{OT}^{t,n}$-hybrid model (*i.e.* assuming access to t-out-of-n OT), which can be subsequently reduced to the $\mathcal{F}_{OT}$-hybrid model via standard techniques for obtaining $\mathcal{F}_{OT}^{t,n}$ from $\mathcal{F}_{OT}$ [Nao91,BCR86,NP99]. We denote by $\mathcal{F}_{OT}^{t,n}(\lambda)$ an instance

[4] Our work has been recognised by the authors of [GIKW14] as being independent.

Functionality $\mathcal{F}_{\text{HCOM}}$

$\mathcal{F}_{\text{HCOM}}$ proceeds as follows, running with parties $P_1, \ldots, P_n$ and an adversary S:

- **Commit Phase**: Upon receiving a message $(\mathsf{commit}, sid, ssid, P_s, P_r, \boldsymbol{m})$ from P_s, where $\boldsymbol{m} \in \{0,1\}^\lambda$, record the tuple $(ssid, P_s, P_r, \boldsymbol{m})$ and send the message $(\mathsf{receipt}, sid, ssid, P_s, P_r)$ to P_r and S. (The lengths of the strings λ is fixed and known to all parties.) Ignore any future commit messages with the same $ssid$ from P_s to P_r. If a message $(\mathsf{abort}, sid, ssid)$ is received from S, the functionality halts.
- **Reveal Phase**: Upon receiving a message $(\mathsf{reveal}, sid, ssid)$ from P_s: If a tuple $(ssid, P_s, P_r, \boldsymbol{m})$ was previously recorded, then send the message $(\mathsf{reveal}, sid, ssid, P_s, P_r, \boldsymbol{m})$ to P_r and S. Otherwise, ignore.
- **Addition**: Upon receiving a message $(\mathsf{add}, sid, ssid, P_s, ssid_1, ssid_2, ssid_3)$ from P_r: If tuples $(ssid_1, P_s, P_r, \boldsymbol{m}_1)$, $(ssid_2, P_s, P_r, \boldsymbol{m}_2)$ were previously recorded and $ssid_3$ is unused, record $(ssid_3, P_s, P_r, \boldsymbol{m}_1 + \boldsymbol{m}_2)$ and send the message $(\mathsf{add}, sid, ssid, P_s, ssid_1, ssid_2, ssid_3, \mathsf{success})$ to P_s, P_r and S.
- **Multiplication**: Upon receiving a message $(\mathsf{mult}, sid, ssid, P_s, ssid_1, ssid_2, ssid_3)$ from P_r: If tuples $(ssid_1, P_s, P_r, \boldsymbol{m}_1)$, $(ssid_2, P_s, P_r, \boldsymbol{m}_2)$ and $(ssid_3, P_s, P_r, \boldsymbol{m}_3)$ were previously recorded, and if $\boldsymbol{m}_3 = \boldsymbol{m}_1 * \boldsymbol{m}_2$, send the message $(\mathsf{mult}, sid, ssid, P_s, ssid_1, ssid_2, ssid_3, \mathsf{success})$ to P_s, P_r and S. Otherwise, send message $(\mathsf{mult}, sid, ssid, P_s, ssid_1, ssid_2, ssid_3, \mathsf{fail})$ to P_s, P_r and S.
- **Linear Function Evaluation:** Upon receiving a message $(\mathsf{linear}, sid, ssid, P_s, \varphi, ssid_1, ssid_2)$, where φ is a linear function, from P_s: If the tuple $(ssid_1, P_s, P_r, \boldsymbol{m}_1)$ was previously recorded and $ssid_2$ is unused, store $(ssid_2, P_s, P_r, \varphi(\boldsymbol{m}_1))$ and send $(\mathsf{linear}, sid, ssid, P_s, ssid_1, ssid_2, \mathsf{success})$ to P_s, P_r and S.

Fig. 1. Functionality $\mathcal{F}_{\text{HCOM}}$

Functionality $\mathcal{F}_{OT}^{t,n}$

$\mathcal{F}_{OT}^{t,n}$ interacts with a sender P_s, a receiver P_r and an adversary S.

- Upon receiving a message $(\mathsf{sender}, sid, ssid, \boldsymbol{x}_0, \ldots, \boldsymbol{x}_n)$ from P_s, where each $\boldsymbol{x}_i \in \{0,1\}^\lambda$, store the tuple $(ssid, \boldsymbol{x}_0, \ldots, \boldsymbol{x}_n)$ (The lengths of the strings λ is fixed and known to all parties). Ignore further messages from P_s to P_r with the same $ssid$.
- Upon receiving a message $(\mathsf{receiver}, sid, ssid, c_1, \ldots, c_t)$ from P_r, check if a tuple $(ssid, \boldsymbol{x}_0, \ldots, \boldsymbol{x}_n)$ was recorded. If yes, send $(\mathsf{received}, sid, ssid, \boldsymbol{x}_{c_1}, \ldots, \boldsymbol{x}_{c_t})$ to P_r and $(\mathsf{received}, sid, ssid)$ to P_s and halt. If not, send nothing to P_r (but continue running).

Fig. 2. Functionality $\mathcal{F}_{OT}^{t,n}$

of the functionality that takes as input from the sender messages in $\{0,1\}^\lambda$. Notice that $\mathcal{F}_{OT}$ can be efficiently UC-realized by the protocols in [PVW08], which can be used to instantiate our commitment protocol. We define our commitment functionality $\mathcal{F}_{\text{HCOM}}$ in Figure 1 and $\mathcal{F}_{OT}^{t,n}$ in Figure 2, further definitions can be found in the full version of this paper [DDGN14].

2.2 Linear Secret Sharing

In very short terms, a linear secret sharing scheme is a secret sharing scheme defined over a finite field $\mathbb{F}$, where the shares are computed as a linear function of the secret (consisting of one or more field elements) and some random field elements. A special case is Shamir's well known scheme. However, we need a more general model for our purposes. We follow the approach from [CDP12] and recall the definitions we need from their model.

Definition 1. *A linear secret sharing scheme $\mathcal{S}$ over the finite field $\mathbb{F}$ is defined by the following parameters: number of players n, secret length ℓ, randomness length e, privacy threshold t and reconstruction threshold r. Also, a $n \times (\ell + e)$ matrix M over $\mathbb{F}$ is given and $\mathcal{S}$ must have r-reconstruction and t-privacy as explained below. If $\ell > 1$, then $\mathcal{S}$ is called a* packed *linear secret sharing scheme.*

Let $d = \ell + e$ and let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of players, then the row number i of M, denoted by $\boldsymbol{m}_i$, is assigned to player P_i. If A is a player subset, then M_A denotes the matrix consisting of rows from M assigned to players in A. To share a secret $\boldsymbol{s} \in \mathbb{F}^\ell$, one first forms a column vector $\boldsymbol{f} \in \mathbb{F}^d$ where $\boldsymbol{s}$ appears in the first ℓ entries and with the last e entries chosen uniformly at random. The *share vector* of $\boldsymbol{s}$ in the scheme $\mathcal{S}$ is computed as $\boldsymbol{c} = M \cdot \boldsymbol{f}$ and its i-th component $\boldsymbol{c}[i]$ is the share given to the player P_i. We will use π_ℓ to denote the projection that outputs the first ℓ coordinates of a vector, *i.e.* $\pi_\ell(\boldsymbol{f}) = \boldsymbol{s}$.

Now, *t-privacy* means that for any player subset A of size at most t, the distribution of $M_A \cdot \boldsymbol{f}$ is independent of $\boldsymbol{s}$. It is easy to see that this is the case if and only if there exists, for each position j in $\boldsymbol{s}$, a *sweeping vector* $\boldsymbol{w}^{A,j}$. This is a column vector of d components such that $M_A \cdot \boldsymbol{w}^{A,j} = \boldsymbol{0}$ and $\pi_\ell(\boldsymbol{w}^{A,j})$ is a vector whose j-th entry is 1 while all other entries are 0.

Finally, *r-reconstruction* means that for any player subset B of size at least r, $\boldsymbol{s}$ is uniquely determined from $M_B \cdot \boldsymbol{f}$. It is easy to see that this is the case if and only if there exists, for each position j in $\boldsymbol{s}$, a *reconstruction vector* $\boldsymbol{r}_{B,j}$. This is a row vector of $|B|$ components such that for any $\boldsymbol{f} \in \mathbb{F}^d$, $\boldsymbol{r}_{B,j} \cdot M_B \cdot \boldsymbol{f} = \boldsymbol{f}[j]$, where $\boldsymbol{f}[j]$ is the j-th entry in $\boldsymbol{f}$.

A packed secret sharing scheme was constructed in Franklin and Yung [FY92]. However, to get our results, we will need a scheme that works over constant size fields, such an example can be found in [CDP12].

Multiplying Shares: for $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{F}^k$, where $\boldsymbol{v} \otimes_i \boldsymbol{w} = (\boldsymbol{v}[i]\boldsymbol{w}[j])_{j \neq i}$, the vector $\boldsymbol{v} \otimes \boldsymbol{w} \in \mathbb{F}^{k^2}$ is defined by $\boldsymbol{v} \otimes \boldsymbol{w} = (\boldsymbol{v}[1]\boldsymbol{w}[1], \ldots, \boldsymbol{v}[k]\boldsymbol{w}[k], \boldsymbol{v} \otimes_1 \boldsymbol{w}, \ldots, \boldsymbol{v} \otimes_k \boldsymbol{w})$. If M is the matrix of the linear secret sharing scheme $\mathcal{S}$, we can define a new scheme $\widehat{\mathcal{S}}$ considering the matrix $\widehat{M}$, whose i-th row is the vector $\boldsymbol{m}_i \otimes \boldsymbol{m}_i$. Clearly $\widehat{M}$ has n rows and d^2 columns and for any $\boldsymbol{f}^1, \boldsymbol{f}^2 \in \mathbb{F}^d$ it holds that

$(M \cdot \boldsymbol{f}^1) * (M \cdot \boldsymbol{f}^2) = \widehat{M} \cdot (\boldsymbol{f}^1 \otimes \boldsymbol{f}^2)^\top$ where $*$ is just the Schur product (or componentwise product). Note that if t is the privacy threshold of $\mathcal{S}$, then the scheme $\widehat{\mathcal{S}}$ also has the t-privacy property. But in general it does not hold that the $\widehat{\mathcal{S}}$ has r-reconstruction. However, suppose that $\widehat{\mathcal{S}}$ has $(n-t)$-reconstruction, then $\mathcal{S}$ is said to have the *t-strong multiplication property.*

In particular, if $\mathcal{S}$ has the t-strong multiplication property, then for any player set A of size at least $n-t$ and for any index $j = 1, \dots, \ell$ there exists a row vector $\widehat{\boldsymbol{r}}_{A,j}$ such that $\widehat{\boldsymbol{r}}_{A,j} \cdot \left[(M_A \cdot \boldsymbol{f}^1) * (M_A \cdot \boldsymbol{f}^2)\right] = \boldsymbol{s}^1[j]\boldsymbol{s}^2[j]$ for any $\boldsymbol{s}^1, \boldsymbol{s}^2 \in \mathbb{F}^\ell$.

3 Packed Verifiable Secret-Sharing

In a *Verifiable Secret-Sharing* scheme (VSS) a dealer distributes shares of a secret to the players in $\mathcal{P}$ in such a way that the honest players are guaranteed to get consistent shares of a well-defined secret or agree that the dealer cheated. In this section we present a packed verifiable secret sharing protocol that generalizes and combines the ideas of packed secret sharing from [FY92] and VSS based on polynomials in 2 variables from [BOGW88]. The protocol is not a full-blown VSS, as it aborts as soon as anyone complains, but this is all we need for our results. The proofs for all lemmas in this section can be found in the full version of this paper [DDGN14].

The protocol can be based on any linear secret-sharing scheme $\mathcal{S}$ over $\mathbb{F}$ as defined in Section 2. We assume an active adversary who corrupts t players and possibly the dealer, and we assume that at least r players are honest. The protocol will secret-share ℓ column vectors $\boldsymbol{s}^1, \dots, \boldsymbol{s}^\ell \in \mathbb{F}^\ell$. In the following, F will be a $d \times d$ matrix with entries in $\mathbb{F}$ and for $1 \leq i \leq n$ we will define $\boldsymbol{h}^i = F \cdot \boldsymbol{m}_i^\top$ and $\boldsymbol{g}_i = \boldsymbol{m}_i \cdot F$. It is then clear that $\boldsymbol{m}_j \cdot \boldsymbol{h}^i = \boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top$ for $1 \leq i, j \leq n$. We will use $\boldsymbol{f}^b$ to denote the b-th column of F. The protocol is shown in Figure 3.

Packed Verifiable Secret-Sharing Protocol π_{VSS}

1. Let $\boldsymbol{s}^1, \dots, \boldsymbol{s}^\ell \in \mathbb{F}^\ell$ be the secrets to be shared. The dealer chooses a random $d \times d$ matrix F with entries in $\mathbb{F}$, subject to $\pi_\ell(\boldsymbol{f}^b) = \boldsymbol{s}^b$ for any $b = 1, \dots, \ell$.
2. The dealer sends $\boldsymbol{h}^i$ and $\boldsymbol{g}_i$ to P_i.
3. Each player P_j sends $\boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top$ to P_i, for $i = 1, \dots, n$.
4. Each P_i checks, for $j = 1, \dots, n$, that $\boldsymbol{m}_j \cdot \boldsymbol{h}^i$ equals the value received from P_j. He broadcasts *Accept* if all checks are OK, otherwise he broadcasts *Reject*.
5. If all players said *Accept*, then each P_j stores, for $b = 1, \dots, \ell$, $\boldsymbol{g}_j[b]$ as his share of $\boldsymbol{s}^b$, otherwise the protocol aborts.

Fig. 3. The VSS protocol

For a column vector $\boldsymbol{v} \in \mathbb{F}^d$, we will say that $\boldsymbol{v}$ *shares* $\boldsymbol{s} \in \mathbb{F}^\ell$, if $\pi_\ell(\boldsymbol{v}) = \boldsymbol{s}$ and each honest player P_j holds $\boldsymbol{m}_j \cdot \boldsymbol{v}$. In other words, $\boldsymbol{c} = M \cdot \boldsymbol{v}$ forms a share vector of $\boldsymbol{s}$ in exactly the way we defined in the previous section. We now show some basic facts about π_{VSS}:

Lemma 1 (completeness). *If the dealer in π_{VSS} is honest, then all honest players accept and the column vector $\boldsymbol{f}^b$ shares $\boldsymbol{s}^b$ for any $b = 1, \ldots, \ell$.*

Lemma 2 (soundness). *If the dealer in π_{VSS} is corrupt, but no player rejects, then for $b = 1, \ldots, \ell$, there exists a column vector $\boldsymbol{v}^b$ and a bit string $\boldsymbol{s}^b$ such that $\boldsymbol{v}^b$ shares $\boldsymbol{s}^b$.*[5]

Finally, we show a strong privacy property guaranteeing that if we open any linear function of $\boldsymbol{s}^b$'s, then no further information on the $\boldsymbol{s}^b$'s is released. To be more precise about this, assume $T : \mathbb{F}^\ell \mapsto \mathbb{F}^{\ell'}$, where $\ell' \leq \ell$, is a surjective linear function. By $T(\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell)$, we mean a tuple $(\boldsymbol{u}^1, \ldots, \boldsymbol{u}^{\ell'})$ of column vectors in $\mathbb{F}^\ell$ s.t. $\boldsymbol{u}^b[a] = T(\boldsymbol{s}^1[a], \ldots, \boldsymbol{s}^\ell[a])[b]$. Put differently, if we arrange the column vectors $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell$ in a $\ell \times \ell$ matrix, then what happens is that we apply T to each row, and let the $\boldsymbol{u}^b$'s be the columns in the resulting matrix. In a completely similar way, we define a tuple of ℓ' column vectors of length d by the formula $T(\boldsymbol{f}^1, \ldots, \boldsymbol{f}^\ell) = (\boldsymbol{w}^1, \ldots, \boldsymbol{w}^{\ell'})$. It is easy to see that if $\boldsymbol{f}^1, \ldots, \boldsymbol{f}^\ell$ share $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell$, then $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^{\ell'}$ share $\boldsymbol{u}^1, \ldots, \boldsymbol{u}^{\ell'}$, since the players can apply T to the shares they received in the first place, to get shares of $\boldsymbol{u}^1, \ldots, \boldsymbol{u}^{\ell'}$. In the following we will abbreviate and use $T(F)$ to denote $T(\boldsymbol{f}^1, \ldots, \boldsymbol{f}^\ell)$.

Now, by *opening* $T(\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell)$, we mean that the (honest) dealer makes $T(F)$ public, which allows anyone to compute $T(\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell)$. We want to show that, in general, if $T(\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell)$ is opened, then the adversary learns $T(\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell)$ and no more information about $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell$. This is captured by Lemma 3. Suppose that $A = \{P_{i_1}, \ldots, P_{i_t}\}$ is a set of players corrupted by the adversary.

Lemma 3 (privacy). *Suppose the dealer in π_{VSS} is honest. Now, in case 1 suppose he executes π_{VSS} with input $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell$ and then opens $T(\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell)$. In case 2, he executes π_{VSS} with input $\widetilde{\boldsymbol{s}}^1, \ldots, \widetilde{\boldsymbol{s}}^\ell$ and then opens $T(\widetilde{\boldsymbol{s}}^1, \ldots, \widetilde{\boldsymbol{s}}^\ell)$. If $T(\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell) = T(\widetilde{\boldsymbol{s}}^1, \ldots, \widetilde{\boldsymbol{s}}^\ell)$, then the views of the adversary in the two cases are identically distributed.*

As the last step, we show an extra randomness property satisfied by the share vectors obtained by Protocol π_{VSS}. If C is a $a \times b$ matrix, define $\pi^\ell(C)$ as the $a \times \ell$ matrix given by the first ℓ columns of C and $\pi_\ell(C)$ as the $\ell \times b$ matrix given by the first ℓ rows of C. Note that, if V is a $d \times \ell$ matrix such that $\pi_\ell(V) = (\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell)$, then the dealer might have chosen V as the first ℓ columns in his matrix F. We want to show that given the adversary's view, any V could have been chosen, as long as it is consistent with the adversary's shares of $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell$.

Lemma 4 (randomness of the share vectors). *Suppose that the dealer in π_{VSS} is honest and let $A = \{P_{i_1}, \ldots, P_{i_t}\}$ be a set of players corrupted by the adversary. If we define G_A as the matrix whose j-th row is $\boldsymbol{g}_{i_j}$, then all the $d \times \ell$ matrices V such that $\pi_\ell(V) = (\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell)$ and $M_A \cdot V = \pi^\ell(G_A)$ are equally likely, even given the adversary's entire view.*

[5] Recall that this just means that $\pi_\ell(\boldsymbol{v}^b) = \boldsymbol{s}^b$ and secret sharing with $\boldsymbol{v}^b$ produces the shares held by the honest parties in the protocol, *i.e.*, $(M\boldsymbol{v}^b)[j] = \boldsymbol{g}_j[b]$ for all honest P_j.

For the applications of π_{VSS} that we will show in Section 5, we will require some new specialized forms of π_{VSS}, which we describe in the following two sections.

3.1 Applying a Linear Map to All the Secrets

Let $\varphi : \mathbb{F}^\ell \to \mathbb{F}^\ell$ be a linear function. Suppose that the dealer executes two correlated instances of Protocol π_{VSS} in the following way: first the dealer executes π_{VSS} with input $\boldsymbol{s}^1, \dots, \boldsymbol{s}^\ell$ choosing matrix F in step 1, later on, he executes π_{VSS} with input $\varphi(\boldsymbol{s}^1), \dots, \varphi(\boldsymbol{s}^\ell)$ under the condition that the matrix chosen for the second instance, F_φ, satisfies $\pi_\ell(\boldsymbol{f}^{\varphi,i}) = \varphi(\pi_\ell(\boldsymbol{f}^i))$ for $i = 1, \dots, d$. The dealer sends to P_i vectors $\boldsymbol{h}^i$ and $\boldsymbol{g}_i$ and also the vectors $\boldsymbol{h}^{\varphi,i} = F_\varphi \cdot \boldsymbol{m}_i^\top$, $\boldsymbol{g}_{\varphi,i} = \boldsymbol{m}_i \cdot F_\varphi$. The protocol is shown in figure 4.

Packed Verifiable Secret-Sharing Protocol for φ, π_{VSS}^{φ}

1. Let $\boldsymbol{s}^1, \dots, \boldsymbol{s}^\ell \in \mathbb{F}^\ell$ be the secrets to be shared. The dealer chooses two random $d \times d$ matrices F, F_φ subject to $\pi_\ell(\boldsymbol{f}^b) = \boldsymbol{s}^b$ for any $b = 1, \dots, \ell$ and $\pi_\ell(\boldsymbol{f}^{\varphi,i}) = \varphi(\pi_\ell(\boldsymbol{f}^i))$ for any $i = 1, \dots, d$.
2. The dealer sends $\boldsymbol{h}^i$, $\boldsymbol{g}_i$, $\boldsymbol{h}^{\varphi,i}$ and $\boldsymbol{g}_{\varphi,i}$ to P_i.
3. Each player P_j sends $\boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top$ and $\boldsymbol{g}_{\varphi,j} \cdot \boldsymbol{m}_i^\top$ to P_i, for $i = 1, \dots, n$.
4. Each P_i checks, for $j = 1, \dots, n$, that $\boldsymbol{m}_j \cdot \boldsymbol{h}^i$ and $\boldsymbol{m}_j \cdot \boldsymbol{h}^{\varphi,i}$ are equal to the values received from P_j and also that $\pi_\ell(\tilde{\boldsymbol{h}}^i) = \varphi\left(\pi_\ell(\boldsymbol{h}^i)\right)$. He broadcasts *Accept* if all checks are OK, otherwise he broadcasts *Reject*.
5. If all players said *Accept*, then each P_j stores, for $b = 1, \dots, \ell$, $\boldsymbol{g}_j[b]$ and $\boldsymbol{g}_{\varphi,j}[b]$ as his share respectively of $\boldsymbol{s}^b$ and $\varphi(\boldsymbol{s}^b)$, otherwise the protocol aborts.

Fig. 4. The VSS protocol for φ

The completeness of the π_{VSS}^{φ} protocol is trivial to prove. Moreover we will show in the following lemma 5 and 6, that also the properties of soundness and privacy are still valid for the π_{VSS}^{φ} protocol.

Lemma 5. *If the dealer in π_{VSS}^{φ} is corrupt, but no player rejects, then for any $b = 1, \dots, \ell$ there exist column vectors $\boldsymbol{v}^b$, $\boldsymbol{v}^{\varphi,b}$ and $\boldsymbol{s}^b$, $\boldsymbol{s}^{\varphi,b}$ such that $\boldsymbol{v}^b$ shares $\boldsymbol{s}^b$, $\boldsymbol{v}^{\varphi,b}$ shares $\boldsymbol{s}^{\varphi,b}$ and $\varphi\left(\boldsymbol{s}^b\right) = \boldsymbol{s}^{\varphi,b}$.*

Lemma 6. *Suppose the dealer in π_{VSS}^{φ} is honest. Now, in case 1 suppose the dealer executes π_{VSS}^{φ} with input $\boldsymbol{s}^1, \dots, \boldsymbol{s}^\ell$ and in case 2, he executes π_{VSS}^{φ} with input $\widetilde{\boldsymbol{s}}^1, \dots, \widetilde{\boldsymbol{s}}^\ell$. Let $A = \{P_{i_1}, \dots, P_{i_t}\}$ be a set of players corrupted by the adversary, then the adversary's view in the two cases are identically distributed.*

Finally we show the randomness property satisfied by the pair of share vectors of $\boldsymbol{s}^i$, $\varphi(\boldsymbol{s}^i)$ obtained by the π_{VSS}^{φ} protocol.

Lemma 7. *Suppose that the dealer in π_{VSS}^{φ} is honest and let $A = \{P_{i_1}, \dots, P_{i_t}\}$ be a set of players corrupted by the adversary. If we define G_A as the matrix* *whose j-th row is $\boldsymbol{g}_{i_j}$ and $G_{\varphi,A}$ as the matrix whose jth column is $\boldsymbol{g}_{\varphi,i_j}$, then all the pairs of $d \times \ell$ matrices (V, V_φ) such that $\pi_\ell(V) = (\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell)$, $\pi_\ell(V_\varphi) = (\varphi(\boldsymbol{s}^1), \ldots, \varphi(\boldsymbol{s}^\ell))$, $M_A \cdot V = \pi^\ell(G_A)$ and $M_A \cdot V_\varphi = \pi^\ell(G_{\varphi,A})$ are equally likely, even given the adversary's entire view.*

3.2 Sharing an All Zeros Vector

We are interested in modifying Protocol π_{VSS} in order to share several times just the vector 0^ℓ, *i.e.* the all zeros column vector in $\mathbb{F}^\ell$. Suppose that the $d \times d$ random matrix F chosen by the dealer has the first ℓ rows equal to zero. Let R be the $e \times d$ matrix formed by the last e rows of F, then

$$\boldsymbol{h}^i = F \cdot \boldsymbol{m}_i^\top = \left(0, \ldots, 0, R \cdot \boldsymbol{m}_i^\top\right)^\top$$

$$\boldsymbol{g}_i = \boldsymbol{m}_i \cdot F = (\boldsymbol{m}_i[\ell+1], \ldots, \boldsymbol{m}_i[d]) \cdot R$$

Given the special form of the vectors $\boldsymbol{h}^i$, the players can check not only that the shares are consistent, but also that they are consistent with 0^ℓ. Define $\boldsymbol{h}^{\mathbf{0},i} = \left(R \cdot \boldsymbol{m}_i^\top\right)^\top$, $\boldsymbol{m}_{\mathbf{0},i} = (\boldsymbol{m}_i[\ell+1], \ldots, \boldsymbol{m}_i[d])$ and $M_{\mathbf{0},A}$ as the matrix whose rows are the vectors $\boldsymbol{m}_{\mathbf{0},i}$ with $P_i \in A$. The protocol in this case is shown in Figure 5.

Packed Verifiable Secret-Sharing Protocol for 0^ℓ's $\pi^{\mathbf{0}}_{VSS}$

1. The dealer chooses a random $e \times d$ matrix R with entries in $\mathbb{F}$,
2. The dealer sends $\boldsymbol{h}^{\mathbf{0},i}$ and $\boldsymbol{g}_i$ to P_i.
3. Each player P_j sends $\boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top$ to P_i, for $i = 1, \ldots, n$.
4. Each P_i checks that for $j = 1, \ldots, n$, $\boldsymbol{m}_{\mathbf{0},j} \cdot \boldsymbol{h}^{\mathbf{0},i}$ equals the value received from P_j. He broadcasts *Accept* if all checks are OK, otherwise he broadcasts *Reject*.
5. If all players said *Accept*, then each P_j stores, for $b = 1, \ldots, \ell$, $\boldsymbol{g}_j[b]$ as his b-th share of 0^ℓ, otherwise the protocol aborts.

Fig. 5. The VSS protocol for 0^ℓ's

Again the completeness of Protocol $\pi^{\mathbf{0}}_{VSS}$ is trivial. We will show the soundness property in Lemma 8, while privacy is not required in this special case.

Lemma 8. *If the dealer in $\pi^{\mathbf{0}}_{VSS}$ is corrupt, but no player rejects, then there exist column vectors $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^\ell$ each of which shares 0^ℓ.*

Finally we show that the randomness property that is satisfied by the share vectors obtained in Protocol π_{VSS} is also satisfied by the share vectors of 0^ℓ obtained by Protocol $\pi^{\mathbf{0}}_{VSS}$.

Lemma 9. *Suppose that the dealer is honest and he executes Protocol $\pi^{\mathbf{0}}_{VSS}$. Let $A = \{P_{i_1}, \ldots, P_{i_t}\}$ be a set of players corrupted by the adversary and define G_A as the matrix whose j-th row is $\boldsymbol{g}_{i_j}$, then all the $e \times \ell$ matrices V such that $M_{\mathbf{0},A} \cdot V = \pi^\ell(G_A)$ are equally likely, even given the adversary's entire view.*

4 Low Overhead UC Commitments

In this section we introduce our construction of UC commitments with low overhead. A main ingredient will be the n-player VSS scheme from the previous section. We will use n as the security parameter. We will assume throughout that the underlying linear secret sharing scheme $\mathcal{S}$ is such that the parameters t and r are $\Theta(n)$, and furthermore that $\mathcal{S}$ has t-strong multiplication. We will call such an $\mathcal{S}$ a *commitment-friendly* linear secret sharing scheme.

The protocol first does a set-up phase where the sender executes the VSS scheme "in his head", where the secrets are random strings $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_\ell$. The VSS is secure against t corrupted players. Next, he chooses n seeds $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ for a pseudorandom generator G, and $\mathcal{F}_{OT}^{t,n}$ is used to transfer a subset of t seeds to the verifier. Finally, the sender sends the view of each virtual VSS player to the receiver, encrypted with $G(\boldsymbol{x}_1), \ldots, G(\boldsymbol{x}_n)$ as "one-time pads". Note that the receiver can decrypt t of these views and check that they are consistent, and also he now knows t shares of each $\boldsymbol{r}_i$.

To commit to $\boldsymbol{m} \in \{0,1\}^\ell$, the sender picks the next unused secret $\boldsymbol{r}_\eta$ and sends $\boldsymbol{m} + \boldsymbol{r}_\eta$.

To open, the sender reveals $\boldsymbol{m}$ and the vector $\boldsymbol{f}^\eta$ (from the VSS) that shares $\boldsymbol{r}_\eta$. The receiver can now compute all shares of $\boldsymbol{r}_\eta$ and check that they match those he already knows.

Intuitively, this is binding because the sender does not know which VSS players the receiver can watch. This means that the sender must make consistent views for most players, or be rejected immediately. But if most views are consistent, then the (partially encrypted) set of shares of $\boldsymbol{r}_\eta$ that was sent during set-up is almost completely consistent. Since the reconstruction threshold is smaller than n by a constant factor this means that the prover must change many shares to move to a different secret, and the receiver will notice this with high probability, again because the sender does not know which shares are already known to the receiver.

Hiding follows quite easily from security of the PRG G and privacy of the VSS scheme, since the receiver only gets t shares of any secret.

The Commit and Reveal phases of protocol π_{HCOM} are described in Figure 6 while the necessary steps for addition, multiplication and linear function evaluation are described separately in Section 5 for the sake of clarity.

The proof of the following theorem can be found in the full version of this paper [DDGN14].

Theorem 4. *Let $G : \{0,1\}^{\ell_{PRG}} \rightarrow \{0,1\}^{2(\ell+e)}$ be a pseudrandom generator and let π_{VSS} be a packed verifiable secret sharing scheme as described in Section 3 with parameters (M, r, t), based on a commitment-friendly secret sharing scheme. Then protocol π_{HCOM} UC-realizes $\mathcal{F}_{\text{HCOM}}$ in the $\mathcal{F}_{OT}^{t,n}(\ell_{PRG})$-hybrid model in the presence of static, active adversaries.*

Complexity. It is evident that in the set-up phase, or later, P_s could execute any number of instances of the VSS and send the resulting views of players encrypted with the seeds $\{\boldsymbol{x}_i\}$, as long as we have a PRG with sufficient stretch. This way we can accommodate as many commitments as we want, while only using the OT-functionality once.[6] Therefore, the amortised cost of a commitment is essentially only what we pay after the OT has been done. We now consider what the cost will be per committed bit in communication and computation. Using the linear secret sharing scheme from [CDP12], we can get a commitment friendly secret sharing scheme over a constant size field, so this means that the communication overhead is constant.

As for computation, under plausible complexity assumptions, there exists a PRG where we pay only a constant number of elementary bit operations per output bit (see, *e.g.*, [VZ12]), so the cost of computing the PRG adds only a constant factor overhead for both parties. As for the computation of P_r, let us consider the set-up phase first. Let C be the set of players watched by P_r, and let G_C, H_C be matrices where we collect the $\boldsymbol{h}^i$ and $\boldsymbol{g}_j$'s they have been assigned. Then what P_r wants to check is that $M_C H_C = G_C M_C^\top$. In [DZ13], a probabilistic method is described for checking such a relation that has complexity $O(n^2)$ field operations and fails with only negligible probability. This therefore also adds only a constant factor overhead because one VSS instance allows commitment to ℓ^2 bits which is $\Theta(n^2)$. Finally, in the reveal phase P_r computes $M\boldsymbol{f}^\eta$ and verifies a few coordinates. If one can check $\Theta(n)$ such commitments simultaneously, the same trick from [DZ13] can be used, and we get an overall constant factor overhead for P_r. We note that checking many commitments in one go is exactly what we need for the application to non-interactive proofs we describe later.

For P_s, using the scheme from [CDP12], there is no way around doing standard matrix products which can be done in $O(n^{2+\sigma})$ complexity for $\sigma < 1$. This gives us overhead n^σ per committed bit.

Finally, if we use instead standard packed secret sharing based on polynomials, the field size must be linear in n, but on the other hand we can use FFT algorithms in our computations. This gives a poly-logarithmic overhead for both players in communication and computation.

5 Homomorphic Properties

In this section, we show how to implement the add, multiply and linear function commands in $\mathcal{F}_{\text{HCOM}}$. As before, we assume a commitment-friendly linear secret sharing scheme $\mathcal{S}$.

We first need some notation: consider a single commitment as we defined it in the previous section and note that the data pertaining to that commitment consists of a vector $\boldsymbol{f}$ and the committed value $\boldsymbol{m}$ held by P_s, whereas P_r holds

[6] It is not hard to see that since a corrupt P_s looses as soon as P_r sees a single inconsistency, P_s cannot get any advantage from executing a VSS after other commitments have been done.

$\boldsymbol{m} + \pi_\ell(\boldsymbol{f})$ as well as a subset of the coordinates of $M\boldsymbol{f}$. We will refer to the vector $\boldsymbol{m} + \pi_\ell(\boldsymbol{f})$ as the *message field* of the commitment.

We will use $\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f})$ as a shorthand for all this data, where the subscript $\mathcal{S}$ refers to the fact that the matrix M of $\mathcal{S}$ defines the relation between the data of P_s and that of P_r. Whenever we write $\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f})$, this should also be understood as stating that the players in fact hold the corresponding data.

The expression $\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f}) + \mathsf{com}_{\mathcal{S}}(\boldsymbol{m}', \boldsymbol{f}')$ means that both players add the corresponding vectors that they hold of the two commitments, and store the result. It is easy to see that we have

$$\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f}) + \mathsf{com}_{\mathcal{S}}(\boldsymbol{m}', \boldsymbol{f}') = \mathsf{com}_{\mathcal{S}}(\boldsymbol{m} + \boldsymbol{m}', \boldsymbol{f} + \boldsymbol{f}')$$

Furthermore, $\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f}) * \mathsf{com}_{\mathcal{S}}(\boldsymbol{m}', \boldsymbol{f}')$ means that the players compute the coordinate-wise product of corresponding vectors they hold and store the result. We have

$$\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f}) * \mathsf{com}_{\mathcal{S}}(\boldsymbol{m}', \boldsymbol{f}') = \mathsf{com}_{\hat{\mathcal{S}}}(\boldsymbol{m} * \boldsymbol{m}', \boldsymbol{f} \otimes \boldsymbol{f}')$$

Note that $\hat{\mathcal{S}}$ appears in the last term. Recall that the coordinates of $\boldsymbol{f} \otimes \boldsymbol{f}'$ are ordered such that indeed the vector $\pi_\ell(\boldsymbol{f}) * \pi_\ell(\boldsymbol{f}')$ appears in the first ℓ coordinates of $\boldsymbol{f} \otimes \boldsymbol{f}'$.

Now, in order to support the additional commands, we will augment the set-up phase of the protocol: in addition to π_{VSS}, P_s will execute $\pi^{\mathbf{0}}_{VSS}$ and π^{φ}_{VSS}. For $\pi^{\mathbf{0}}_{VSS}$ we use $\hat{\mathcal{S}}$ as the underlying linear secret sharing scheme, where the other VSS schemes use $\mathcal{S}$. Furthermore, we need an instance of π^{φ}_{VSS} for each linear function φ we want to support. As before, all the views of the virtual players are sent to P_r encrypted under the seeds x_i. P_r checks consistency of the views as well as the special conditions that honest players check in $\pi^{\mathbf{0}}_{VSS}$ and π^{φ}_{VSS}.

Note that if one instance of π_{VSS} has been executed, this allows us to extract data for ℓ commitments. Likewise, an execution of $\pi^{\mathbf{0}}_{VSS}$ allows us to extract ℓ commitments of form $\mathsf{com}_{\hat{\mathcal{S}}}(0^\ell, \boldsymbol{u})$ for a random $\boldsymbol{u}$, where by default we set the message field to 0. Finally, having executed π^{φ}_{VSS}, we can extract ℓ pairs of form $\mathsf{com}_{\mathcal{S}}(\boldsymbol{r}, \boldsymbol{f}_r), \mathsf{com}_{\mathcal{S}}(\varphi(\boldsymbol{r}), \boldsymbol{f}'_r)$ where $\boldsymbol{r}$ is random such that $\boldsymbol{r} = \pi_\ell(\boldsymbol{f}_r)$ and $\varphi(\boldsymbol{r}) = \pi_\ell(\boldsymbol{f}'_r)$. Again, for these commitments we set the message field to 0. The protocols are shown in Figure 7.

Generalizations In the basic case we are committing to bit strings, and we note that we can trivially get negation of bits using the operations we already have: Given $\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f})$, P_s commits to 1^ℓ so we have $\mathsf{com}_{\mathcal{S}}(1^\ell, \boldsymbol{f}')$, we output $\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f}) + \mathsf{com}_{\mathcal{S}}(1^\ell, \boldsymbol{f}')$ and P_s opens $\mathsf{com}_{\mathcal{S}}(1^\ell, \boldsymbol{f}')$ to reveal 1^ℓ.

If we do the protocol over a larger field than $\mathbb{F}_2$, it makes sense to also consider multiplication of a commitment by a public constant. This is trivial to implement, both parties simply multiply their respective vectors by the constant.

Proof intution. The protocol in Figure 7 can be proven secure by essentially the same techniques we used for the basic commitment protocol, but we need in addition the specific properties of π_{VSS}, $\pi^{\mathbf{0}}_{VSS}$ and π^{φ}_{VSS}. First of all, it is clear that in the case when the sender is corrupted and the receiver is honest, a simulator for this protocol can extract the messages (and share vectors) in the commitments by following the same procedure as the simulator for the basic commitment protocol. The specific properties of the VSS protocols π_{VSS}, $\pi^{\mathbf{0}}_{VSS}$ and π^{φ}_{VSS} come into play when constructing a simulator for the case when the sender is honest and the receiver is corrupted. More details are presented in the full version of this paper [DDGN14].

6 Applications

Two-party One-sided Functionalities. In this section we consider applications of our implementation of $\mathcal{F}_{\text{HCOM}}$. We will implement a one-sided functionality where only one party P_s has input x and some verifier is to receive output y, where $y = C(x)$ for a Boolean circuit C.

The basic idea of this is straightforward: P_s commits to each bit in x and to each output from a gate in C that is produced when x is the input. Now we can use the commands of $\mathcal{F}_{\text{HCOM}}$ to verify for each gate that the committed output is the correct function of the inputs. Finally, P_s opens the commitment to the final output to reveal y.

However, we would like to exploit the fact that our commitments can contain ℓ-bit strings and support coordinate-wise operations on ℓ-bit strings in parallel. To this end, we can exploit the construction found in [DIK10] (mentioned in the introduction), that allows us to construct from C a new circuit C' computing the same function as C, but where C' can be computed using only operations in parallel on ℓ-bit blocks as well as $\log \ell$ different permutations of the bits in a block. The construction always works, but if C is well-formed, C' will be of size $O(\log(|C|)|C|)$.

With these observations, we can use $\mathcal{F}_{\text{HCOM}}$ operations to compute C' instead of C. The difference to the first simplistic idea is that now every position in a block is used for computation. Hence, the final protocol is non-interactive, assuming the very first step doing the OT has been done. This is because P_s, since he knows C, can predict which multiplications and permutation operations P_r will need to verify, so he can compute the required opening information for commitments and send them immediately. Moreover, if we use the linear secret sharing scheme from [CDP12] as the basis for commitments, then the size of the entire proof as well as of the verifier's computation will be of size $O(|C| \log |C|)$ for well formed circuits. If C is regular we will get complexity $O(|C|))$, since we can implement the rerouting between layers by evaluating the block functions directly. Thus we get the results claimed in Theorem 1.

Multiparty Computation. Due to space constraints the material on MPC based on our VSS (Theorem 2) is left for the full version of this paper [DDGN14].

Protocol π_{HCOM} in the $\mathcal{F}_{OT}^{t,n}(\ell_{PRG})$-hybrid model

Let $G : \{0,1\}^{\ell_{PRG}} \rightarrow \{0,1\}^{2(\ell+e)}$ be a pseudorandom generator and let π_{VSS} be a packed verifiable secret sharing scheme as described in Section 3 with parameters (M, r, t) based on a commitment-friendly linear secret sharing scheme. A sender P_s and a receiver P_r interact between themselves and with $\mathcal{F}_{OT}^{t,n}(\ell_{PRG})$ as follows:

Setup Phase: At the beginning of the protocol P_s and P_r perform the following steps and then wait for inputs.

1. For $i = 1, \ldots, n$, P_s uniformly samples a random string $\boldsymbol{x}_i \in \{0,1\}^{\ell_{PRG}}$. P_s sends $(\mathsf{sender}, sid, ssid, x_1, \ldots, x_n)$ to $\mathcal{F}_{OT}^{t,n}(\ell_{PRG})$.
2. P_r uniformly samples a set of t indexes $c_1, \ldots, c_t \leftarrow [1, n]$ and sends $(\mathsf{receiver}, sid, ssid, c_1, \ldots, c_t)$ to $\mathcal{F}_{OT}^{t,n}$.
3. Upon receiving $(\mathsf{received}, sid, ssid)$ from $\mathcal{F}_{OT}^{t,n}$, P_s uniformly samples n random strings $\boldsymbol{r}_i \leftarrow \{0,1\}^{\ell}$, $i = 1, \ldots, \ell$ and internally runs π_{VSS} using $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_n$ as input, constructing n strings $((\boldsymbol{h}^i)^\top, \boldsymbol{g}_i)$, $i = 1, \ldots, n$ of length $2(\ell + e)$ from the vectors generated by π_{VSS}. P_s computes $((\widetilde{\boldsymbol{h}}^i)^\top, \widetilde{\boldsymbol{g}}_i) = ((\boldsymbol{h}^i)^\top, \boldsymbol{g}_i) + G(\boldsymbol{x}_i)$ and sends $(sid, ssid, ((\widetilde{\boldsymbol{h}}^1)^\top, \widetilde{\boldsymbol{g}}_1), \ldots, ((\widetilde{\boldsymbol{h}}^n)^\top, \widetilde{\boldsymbol{g}}_n))$ to P_r.
4. Upon receiving $(\mathsf{received}, sid, ssid, \boldsymbol{x}_{c_1}, \ldots, \boldsymbol{x}_{c_t})$ from $\mathcal{F}_{OT}^{t,n}$ and $(sid, ssid, ((\widetilde{\boldsymbol{h}}^1)^\top, \widetilde{\boldsymbol{g}}_1), \ldots, ((\widetilde{\boldsymbol{h}}^n)^\top, \widetilde{\boldsymbol{g}}_n))$ from P_s, P_r computes $((\boldsymbol{h}^{c_j})^\top, \boldsymbol{g}_{c_j}) = ((\widetilde{\boldsymbol{h}}^{c_j})^\top, \widetilde{\boldsymbol{g}}_{c_j}) - G(\boldsymbol{x}_{c_j}), 1 \leq j \leq t$ and uses the procedures of π_{VSS} to check that the shares $\boldsymbol{g}_{c_1}, \ldots, \boldsymbol{g}_{c_t}$ are valid, *i.e.* it checks that $\boldsymbol{m}_j \cdot \boldsymbol{h}^i = \boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top$ for $i, j \in \{c_1, \ldots, c_t\}$. If all shares are valid P_r stores $(ssid, sid, ((\boldsymbol{h}^{c_1})^\top, \boldsymbol{g}_{c_1}), \ldots, ((\boldsymbol{h}^{c_t})^\top, \boldsymbol{g}_{c_t}))$, otherwise it halts.

Commit Phase:

1. Upon input $(\mathsf{commit}, sid, ssid, P_s, P_r, \boldsymbol{m})$, P_s chooses an unused[a] random string $\boldsymbol{r}_\eta$, computes $\widetilde{\boldsymbol{m}} = \boldsymbol{m} + \boldsymbol{r}_\eta$ and sends $(sid, ssid, \eta, \widetilde{\boldsymbol{m}})$ to P_r.
2. P_r stores $(sid, ssid, \widetilde{\boldsymbol{m}})$ and outputs $(\mathsf{receipt}, sid, ssid, P_s, P_r)$.

Reveal Phase:

1. Upon input $(\mathsf{reveal}, sid, ssid, P_s, P_r)$, to reveal a message $\boldsymbol{m}$, P_s reveals the random string $\boldsymbol{r}_\eta$ by sending $(sid, ssid, \boldsymbol{m}, \boldsymbol{f}^\eta)$ to P_r.[b]
2. P_r receives $(sid, ssid, \eta, \boldsymbol{m}, \boldsymbol{f}^\eta)$, computes $M\boldsymbol{f}^\eta = (\overline{\boldsymbol{g}}_1[\eta], \ldots, \overline{\boldsymbol{g}}_n[\eta])^\top$, checks that $\boldsymbol{g}_j[\eta] = \overline{\boldsymbol{g}}_j[\eta]$ for $j \in \{c_1, \ldots, c_t\}$ and that $\boldsymbol{m} = \widetilde{\boldsymbol{m}} - \boldsymbol{r}_\eta$. If the shares pass this check, P_r outputs $(\mathsf{reveal}, sid, ssid, P_s, P_r, \boldsymbol{m})$. Otherwise, it rejects the commitment and halts.

[a] We say that a string $\boldsymbol{r}_\eta$ is unused if it has not been selected by P_s for use in any previous commitment.

[b] Recall that $\boldsymbol{f}^\eta$ denotes the η-th column of F, $\pi_\ell(\boldsymbol{f}^\eta) = \boldsymbol{r}_\eta$ and that $M\boldsymbol{f}^\eta = (\boldsymbol{g}_1[\eta], \ldots, \boldsymbol{g}_n[\eta])^\top$, *i.e.* $\boldsymbol{f}^\eta$ determines the shares of $\boldsymbol{r}_\eta$ generated in the setup phase.

Fig. 6. Protocol π_{HCOM} in the $\mathcal{F}_{OT}^{t,n}(\ell_{PRG})$-hybrid model

Protocols for addition, multiplication and linear operations

Setup Phase: Is augmented by executions of $\pi^{\mathbf{0}}_{VSS}$ and π^{φ}_{VSS} as described in the text. Throughout, opening a commitment $\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f})$ means that P_s sends $\boldsymbol{m}, \boldsymbol{f}$ and P_r verifies, as in π_{HCOM}.

Addition: Given commitments $\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f}), \mathsf{com}_{\mathcal{S}}(\boldsymbol{m}', \boldsymbol{f}')$, output

$$\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f}) + \mathsf{com}_{\mathcal{S}}(\boldsymbol{m}', \boldsymbol{f}') = \mathsf{com}_{\mathcal{S}}(\boldsymbol{m} + \boldsymbol{m}', \boldsymbol{f} + \boldsymbol{f}').$$

Multiplication: Given commitments $\mathsf{com}_{\mathcal{S}}(\boldsymbol{a}, \boldsymbol{f}_a), \mathsf{com}_{\mathcal{S}}(\boldsymbol{b}, \boldsymbol{f}_b)$, and $\mathsf{com}_{\mathcal{S}}(\boldsymbol{c}, \boldsymbol{f}_c)$ extract the next unused commitment from $\pi^{\mathbf{0}}_{VSS}$, $\mathsf{com}_{\hat{\mathcal{S}}}(0^{\ell}, \boldsymbol{u})$. Form a default commitment $\mathsf{com}_{\mathcal{S}}(1^{\ell}, \boldsymbol{f}_1)$, where $\pi_{\ell}(\boldsymbol{f}_1) = 1^{\ell}$ and the other coordinates are 0. This can be done by only local computation. P_s opens the following commitment to reveal 0^{ℓ}:

$$\mathsf{com}_{\mathcal{S}}(\boldsymbol{a}, \boldsymbol{f}_a) * \mathsf{com}_{\mathcal{S}}(\boldsymbol{b}, \boldsymbol{f}_b) - \mathsf{com}_{\mathcal{S}}(\boldsymbol{c}, \boldsymbol{f}_c) * \mathsf{com}_{\mathcal{S}}(1^{\ell}, \boldsymbol{f}_1) + \mathsf{com}_{\hat{\mathcal{S}}}(0^{\ell}, \boldsymbol{u}) = \\ \mathsf{com}_{\hat{\mathcal{S}}}(\boldsymbol{a} * \boldsymbol{b} - \boldsymbol{c}, \boldsymbol{f}_a \otimes \boldsymbol{f}_b - \boldsymbol{f}_c \otimes \boldsymbol{f}_1 + \boldsymbol{u})$$

Linear Function Given commitment $\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f})$, extract from π^{φ}_{VSS} the next unused pair $\mathsf{com}_{\mathcal{S}}(\boldsymbol{r}, \boldsymbol{f}_r), \mathsf{com}_{\mathcal{S}}(\varphi(\boldsymbol{r}), \boldsymbol{f}'_r)$. P_s opens $\mathsf{com}_{\mathcal{S}}(\boldsymbol{m}, \boldsymbol{f}) - \mathsf{com}_{\mathcal{S}}(\boldsymbol{r}, \boldsymbol{f}_r)$ to reveal $\boldsymbol{m} - \boldsymbol{r}$. Both parties compute $\varphi(\boldsymbol{m} - \boldsymbol{r})$ and form a vector $\boldsymbol{v}$ such that $\pi_{\ell}(\boldsymbol{v}) = \varphi(\boldsymbol{m} - \boldsymbol{r})$ and the rest of the entries are 0. Output

$$\mathsf{com}_{\mathcal{S}}(\varphi(\boldsymbol{r}), \boldsymbol{f}'_r) + \mathsf{com}_{\mathcal{S}}(\varphi(\boldsymbol{m} - \boldsymbol{r}), \boldsymbol{v}) = \mathsf{com}_{\mathcal{S}}(\varphi(\boldsymbol{m}), \boldsymbol{f}'_r + \boldsymbol{v})$$

Fig. 7. Protocol for homomorphic operations on commitments

String Commitment with Partial Individual Opening. Here we wish to implement a functionality $F^{N,r}_{com}$ that first allows P_s to commit to N bits and then to open up to r bits individually, where he can decide adaptively which bits to open. We do this in the correlated random bits model where a functionality is assumed that initially gives bit strings to P_s and P_r with some prescribed joint distribution, the implementation must be statistically secure with error probability 2^{-k}.

Note that our protocol can be seen as a protocol in this model if we let players start from the strings that are output by the PRG. In this case we get statistically secure commitments with error probability $2^{-\Theta(\ell)}$ (since ℓ is $\Theta(n)$). So we can choose ℓ to be $\Theta(k)$ and get the required error probability. Then one of our commitments can be realised while consuming $O(k) = O(\ell)$ correlated random bits.

Note that we can open a single bit in a commitment to $\boldsymbol{a}$ as follows: to open the j'th bit a_j the prover commits to $\boldsymbol{e}_j$, a vector with 1 in position j and 0 elsewhere and to $\boldsymbol{c}$ which has a_j in position j and 0's elsewhere. Now the multiplication check is done on commitments to $\boldsymbol{a}, \boldsymbol{e}_j$ and $\boldsymbol{c}$, and P_s opens $\boldsymbol{e}_j$ and $\boldsymbol{c}$. P_r does the obvious checks and extracts a_j. It is trivial to show that this is a secure way to reveal only a_j and we consume $O(\ell)$ correlated random bits since only a constant number of commitments are involved.

Now we can implement $F_{com}^{N,r}$ with $N = \ell^u$ and $r = \ell^{u-1}$ for some u, and the implementation is done by having P_s commit to the N bits in the normal way using r commitments, and when opening any single bit, we execute the above procedure. This consumes a total of $O(N + r\ell) = O(N)$ correlated random bits. Thus the consumption per bit committed to is $O(1)$. Furthermore, we have $r = N^{(u-1)/u} = N^{1-1/u}$, so we get the result of Theorem 3 by choosing a large enough u.

Acknowledgements. We thank Yuval Ishai for pointing out interesting applications of our results and Ignacio Cascudo for clarifying key facts about algebraic geometric secret sharing schemes.

References

App13. Applebaum, B.: Pseudorandom generators with long stretch and low locality from random local one-way functions. SIAM Journal on Computing 42(5), 2008–2037 (2013)

BBDK00. Beimel, A., Burmester, M., Desmedt, Y., Kushilevitz Computing, E.: functions of a shared secret. SIAM J. Discrete Math. 13(3), 324–345 (2000)

BCPV13. Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: Analysis and improvement of lindell's UC-secure commitment schemes. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 534–551. Springer, Heidelberg (2013)

BCR86. Brassard, G., Crepeau, C., Robert, J.-M.: Information theoretic reductions among disclosure problems. In: 27th Annual Symposium on Foundations of Computer Science, pp. 168–173 (1986)

BOGW88. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10 (1988)

Can01. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145. IEEE Computer Society (2001)

CDM00. Cramer, R., Damgård, I., Maurer, U.M.: General secure multi-party computation from any linear secret-sharing scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)

CDP12. Cramer, R., Damgård, I., Pastro, V.: On the amortized complexity of zero knowledge protocols for multiplicative relations. In: Smith, A. (ed.) ICITS 2012. LNCS, vol. 7412, pp. 62–79. Springer, Heidelberg (2012)

CF01. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)

CLOS02. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)

DDGN14. Damgård, I., David, B., Giacomelli, I., Nielsen, J.B.: Compact VSS and efficient homomorphic UC commitments. Cryptology ePrint Archive: Report 2014/370 (2014), `https://eprint.iacr.org/2014/370`

DG03. Damgård, I., Groth, J.: Non-interactive and reusable non-malleable commitment schemes. In: Larmore, L.L., Goemans, M.X. (eds.) STOC, pp. 426–437. ACM (2003)

DI06. Damgård, I.B., Ishai, Y.: Scalable secure multiparty computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 501–520. Springer, Heidelberg (2006)

DIK$^+$08. Damgård, I., Ishai, Y., Krøigaard, M., Nielsen, J.B., Smith, A.: Scalable multiparty computation with nearly optimal work and resilience. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 241–261. Springer, Heidelberg (2008)

DIK10. Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly secure multiparty computation and the computational overhead of cryptography. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 445–465. Springer, Heidelberg (2010)

DNO10. Damgård, I., Nielsen, J.B., Orlandi, C.: On the necessary and sufficient assumptions for UC computation. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 109–127. Springer, Heidelberg (2010)

DZ13. Damgård, I., Zakarias, S.: Constant-overhead secure computation of boolean circuits using preprocessing. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 621–641. Springer, Heidelberg (2013)

FJN$^+$13. Frederiksen, T.K., Jakobsen, T.P., Nielsen, J.B., Nordholt, P.S., Orlandi, C.: Minilego: Efficient secure two-party computation from general assumptions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 537–556. Springer, Heidelberg (2013)

FY92. Franklin, M.K., Yung, M.: Communication complexity of secure computation (extended abstract). In: STOC, pp. 699–710 (1992)

GIKW14. Garay, J., Ishai, Y., Kumaresan, R., Wee, H.: On the complexity of uc commitments. In: EuroCrypt 2014 (to appear 2014)

IKOS08. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Dwork, C. (ed.) STOC, pp. 433–442. ACM (2008)

IKOS09. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge proofs from secure multiparty computation. SIAM J. Comput. 39(3), 1121–1152 (2009)

IPS08. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)

Lin11. Lindell, Y.: Highly-efficient universally-composable commitments based on the ddh assumption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 446–466. Springer, Heidelberg (2011)

Nao91. Naor, M.: Bit commitment using pseudorandomness. J. Cryptology 4(2), 151–158 (1991)

NP99. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999)

PVW08. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)

RTWW11. Ranellucci, S., Tapp, A., Winkler, S., Wullschleger, J.: On the efficiency of bit commitment reductions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 520–537. Springer, Heidelberg (2011)

VZ12. Vadhan, S., Zheng, C.J.: Characterizing pseudoentropy and simplifying pseudorandom generator constructions. In: Proceedings of the 44th symposium on Theory of Computing, pp. 817–836. ACM (2012)