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 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

* 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, within which part of this work was performed; 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 the European Research Commission Starting Grant 279447.

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,

[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

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. 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

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.

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 (see Appendix D for more details).

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 output 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 $\kappa \in \mathbb{N}$, $\mid Pr[D(X_\kappa) = 1] - Pr[D(Y_\kappa) = 1] \mid < \epsilon(\kappa)$. 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 $\mathcal{F}_{\mathrm{HCOM}}$ in Figure 1 and $\mathcal{F}_{OT}^{t,n}$ in Figure 2, further definitions can be found in Appendix A.

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 $\mathcal{P} = \{P_1, \dots, P_n\}$ 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 $\boldsymbol{v} \otimes \boldsymbol{w} = (\boldsymbol{v}[1]\boldsymbol{w}[1], \dots, \boldsymbol{v}[k]\boldsymbol{w}[k], \boldsymbol{v} \otimes_1 \boldsymbol{w}, \dots, \boldsymbol{v} \otimes_k \boldsymbol{w})$. 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

$\left(M \cdot \boldsymbol{f}^1\right) * \left(M \cdot \boldsymbol{f}^2\right) = \widehat{M} \cdot \left(\boldsymbol{f}^1 \otimes \boldsymbol{f}^2\right)^\top$ 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 $j = 1, \ldots, \ell$ there exists a row vector $\widehat{\boldsymbol{r}}_{A,j}$ such that $\widehat{\boldsymbol{r}}_{A,j} \cdot \left[\left(M_A \cdot \boldsymbol{f}^1\right) * \left(M_A \cdot \boldsymbol{f}^2\right)\right] = \boldsymbol{s}^1[j]\boldsymbol{s}^2[j]$ 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 Appendix B.

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 $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell \in \mathbb{F}^\ell$. 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 $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell \in \mathbb{F}^\ell$ 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 $b = 1, \ldots, \ell$.
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 $i = 1, \ldots, n$.
4. Each P_i checks, for $j = 1, \ldots, n$, 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 $b = 1, \ldots, \ell$, $\boldsymbol{g}_j[b]$ 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 need to show privacy. For VSS protocols this is normally just a result saying that if the dealer is honest, then the shares held by corrupt players have distribution independent of the secrets, the $\boldsymbol{s}^b$'s. However, we need in the following a more elaborate result saying 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 $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell$ choosing matrix F in step 1, later on, he executes π_{VSS} with input $\varphi(\boldsymbol{s}^1), \ldots, \varphi(\boldsymbol{s}^\ell)$ 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 $i = 1, \ldots, d$. 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 $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell \in \mathbb{F}^\ell$ 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 $b = 1, \ldots, \ell$ and $\pi_\ell(\boldsymbol{f}^{\varphi,i}) = \varphi(\pi_\ell(\boldsymbol{f}^i))$ for any $i = 1, \ldots, d$.
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 $i = 1, \ldots, n$.
4. Each P_i checks, for $j = 1, \ldots, n$, 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 $b = 1, \ldots, \ell$, $\boldsymbol{g}_j[b]$ 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 $b = 1, \ldots, \ell$ 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 $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell$ and in case 2, he executes π_{VSS}^{φ} with input $\widetilde{\boldsymbol{s}}^1, \ldots, \widetilde{\boldsymbol{s}}^\ell$. Let $A = \{P_{i_1}, \ldots, P_{i_t}\}$ 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 $A = \{P_{i_1}, \ldots, P_{i_t}\}$ be a set of players corrupted by the adversary. If we define G_A as the matrix* *whose j-th row is $\boldsymbol{g}_{i_j}$ 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 Appendix C.

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

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

Let $G : \{0,1\}^{\ell_{PRG}} \to \{0,1\}^{2(\ell+e)}$ 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

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 $\boldsymbol{m} + \pi_\ell(\boldsymbol{f})$ 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.

[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.

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 $\pi_{VSS}^{\mathbf{0}}$ and π_{VSS}^{φ}. For $\pi_{VSS}^{\mathbf{0}}$ 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_{VSS}^{φ} 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 $\pi_{VSS}^{\mathbf{0}}$ and π_{VSS}^{φ}.

Note that if one instance of π_{VSS} has been executed, this allows us to extract data for ℓ commitments. Likewise, an execution of $\pi_{VSS}^{\mathbf{0}}$ 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_{VSS}^{φ}, 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}, $\pi_{VSS}^{\mathbf{0}}$ and π_{VSS}^{φ}. First of all, it is clear that in the case when the sender is corrupted and the receiver is honest,

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.

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.

In the protocol for computing the addition of two commitments, the simulator for a corrupted sender can simply use the same procedure as in the case of the commitment protocol for extracting the messages $\boldsymbol{m}$,$\boldsymbol{m}'$ and proceed using this information. Security then follows from the same arguments as before. In the case when the receiver is corrupted, the simulator can also proceed as the commitment protocol simulator, *i.e.* by committing to random messages and then generating a valid opening to the real messages received from $\mathcal{F}_{\mathrm{HCOM}}$. However, in this case, security follows from Lemma 3, which shows that we can construct a vector $\hat{\boldsymbol{f}}$ corresponding to an arbitrary linear function of $\boldsymbol{f}$ and $\boldsymbol{f}'$ such that the components of $M\hat{\boldsymbol{f}}$ checked by the corrupted receiver are the same as the ones revealed at the setup phase. This vector can be used to simulate the honest P_s opening the sum.

In the protocol for verifying multiplicative relations, the simulator for a corrupted sender can use the same procedure as in the case of the commitment protocol to extract the messages $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, proceeding the simulation with this information. Security follows from the same arguments provided for the simulator for the commitment protocol and also from Lemma 8, which guarantees that

$\mathsf{com}_{\hat{\mathcal{S}}}(0^\ell, \boldsymbol{u})$ is really a commitment to 0^ℓ if the checks in the setup phase are successful. In the case when the receiver is corrupted, the simulator generates commitments to random messages and later on computes valid openings corresponding to the real messages received from $\mathcal{F}_{\mathrm{HCOM}}$. Computing such valid openings for arbitrary messages (in the sense that the components of the share vectors revealed in the setup phase are the same) is guaranteed by Lemma 3 for both π_{VSS} and $\pi^{\mathbf{0}}_{VSS}$.

In the protocol for linear function evaluation, the simulator for a corrupted sender can use the same procedure as in the case of the commitment protocol to extract the message $\boldsymbol{m}$ and proceeds from there. Security in this case follows from the same arguments used for the simulator for the commitment protocol and from Lemma 5, which guarantees that commitments that pass the tests at the setup phase are valid. In the case when the receiver is corrupted, the simulator commits to random messages and later on generates openings to the valid messages received from $\mathcal{F}_{\mathrm{HCOM}}$. Lemma 3 (for π_{VSS}) and Lemma 6 (for π^{φ}_{VSS}) guarantee that it is possible to generate new vectors $\boldsymbol{f}$ (resp. $\boldsymbol{f}'_r$) corresponding to an arbitrary message such that the components of $M\boldsymbol{f}$ (resp. $M\boldsymbol{f}'_r$) revealed in the setup phase are the same.

6 Applications

6.1 Two-party One-sided Functionalities

In this section we consider applications of our implementation of $\mathcal{F}_{\mathrm{HCOM}}$. 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 $\mathcal{F}_{\mathrm{HCOM}}$ 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. We can support both types of operations (since a permutation of coordinates is a linear function) and since we only need a small number of different permutations, the required set-up phase can be very efficient. Since the required permutations do not depend on C, the set-up phase can even be independent of C. But it can also be executed on the fly when C is known, and this will only add a constant factor overhead. The construction always works, but if C is well-formed, C' will be of size $O(\log(|C|)|C|)$. If instead C is regular as defined in the introduction we can instead do the required rerouting of bits between layers by evaluating linear block functions. Since regularity requires that any block function occurring is used several times, we can prepare commitments for checking such linear relations efficiently.

With these observations, we can use $\mathcal{F}_{\mathrm{HCOM}}$ 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. Therefore the protocol implementing this will have several nice properties: first of all, it 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. Second, 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 $O(|C|)$): instead of using [DIK10] we can implement the rerouting between layers by evaluating the block functions directly. This can be done by calling the Linear Function operation from $\mathcal{F}_{\mathrm{HCOM}}$ a constant number of times for each block function. Thus we get the results claimed in Theorem 1.

6.2 Multiparty Computation

Due to space constraints the material on MPC based on our VSS (Theorem 2) is in Appendix E.

6.3 String Commitment with Partial Individual Opening

Here we wish to implement a functionality $F_{com}^{N,r}$ 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] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM Journal on Computing*, 42(5):2008–2037, 2013.

[BBDK00] Amos Beimel, Mike Burmester, Yvo Desmedt, and Eyal Kushilevitz. Computing functions of a shared secret. *SIAM J. Discrete Math.*, 13(3):324–345, 2000.

[BCPV13] Olivier Blazy, Celine Chevalier, David Pointcheval, and Damien Vergnaud. Analysis and improvement of lindells uc-secure commitment schemes. In Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 534–551. Springer Berlin Heidelberg, 2013.

[BCR86] G. Brassard, Claude Crepeau, and J.-M. Robert. Information theoretic reductions among disclosure problems. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 168–173, Oct 1986.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[BTH06] Zuzana Beerliova-Trubiniova and Martin Hirt. Efficient multi-party computation with dispute control. In *Theory of Cryptography*, pages 305–328. Springer, 2006.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.

[CC06] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 521–536. Springer, 2006.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 316–334. Springer, 2000.

[CDP12] Ronald Cramer, Ivan Damgård, and Valerio Pastro. On the amortized complexity of zero knowledge protocols for multiplicative relations. In Adam Smith, editor, *ICITS*, volume 7412 of *Lecture Notes in Computer Science*, pages 62–79. Springer, 2012.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 2001.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.

[DG03] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In Lawrence L. Larmore and Michel X. Goemans, editors, *STOC*, pages 426–437. ACM, 2003.

[DI06] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *Advances in Cryptology-CRYPTO 2006*, pages 501–520. Springer, 2006.

[DI14] Erez Druk and Yuval Ishai. Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 169–182. ACM, 2014.

[DIK$^+$08] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In *Advances in Cryptology–CRYPTO 2008*, pages 241–261. Springer, 2008.

[DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *Proceedings of EuroCrypt*, pages 445–465, Springer Verlag 2010.

[DNO10] Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi. On the necessary and sufficient assumptions for uc computation. In *Theory of Cryptography*, pages 109–127. Springer, 2010.

[DZ13] Ivan Damgård and Sarah Zakarias. Constant-overhead secure computation of boolean circuits using preprocessing. In *Theory of Cryptography*, pages 621–641. Springer, 2013.

[FJN$^+$13] Tore Kasper Frederiksen, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Peter Sebastian Nordholt, and Claudio Orlandi. Minilego: Efficient secure two-party computation from general assumptions. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 537–556. Springer, 2013.

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *STOC*, pages 699–710. ACM, 1992.

[GIKW14] Juan Garay, Yuval Ishai, Ranjit Kumaresan, and Hoeteck Wee. On the complexity of uc commitments. To appear in EuroCrypt 2014, 2014.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Cynthia Dwork, editor, *STOC*, pages 433–442. ACM, 2008.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer–efficiently. In *Advances in Cryptology–CRYPTO 2008*, pages 572–591. Springer, 2008.

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the ddh assumption. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 446–466. Springer, 2011.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In Michael Wiener, editor, *Advances in Cryptology CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 573–590. Springer Berlin Heidelberg, 1999.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer Berlin Heidelberg, 2008.

[RTWW11] Samuel Ranellucci, Alain Tapp, Severin Winkler, and Jürg Wullschleger. On the efficiency of bit commitment reductions. In *Advances in Cryptology–ASIACRYPT 2011*, pages 520–537. Springer, 2011.

[VZ12] Salil Vadhan and Colin Jia Zheng. Characterizing pseudoentropy and simplifying pseudorandom generator constructions. In *Proceedings of the 44th symposium on Theory of Computing*, pages 817–836. ACM, 2012.

A Universal Composability

In this section we will present definitions pertaining to the Universal Composability Framework of [Can01] according to the syntax of [CLOS02] and refer the readers to [Can01,CLOS02] for further definitions and details.

In the UC framework, the entities involved in both the real and ideal world executions are modeled as probabilistic Interactive Turing Machines (ITM) that receive and deliver messages through their input and output tapes, respectively. In the ideal world execution, dummy parties (possibly controlled by an ideal adversary S referred to as the *simulator*) interact directly with the ideal functionality $\mathcal{F}$, which works as a trusted third party that computes the desired primitive. In the real world execution, several parties (possibly corrupted by a real world adversary $\mathcal{A}$) interact with each other by means of a protocol π that realizes the ideal functionality. The real and ideal executions are controlled by the *environment* $\mathcal{Z}$, an entity that delivers inputs and reads the outputs of the individual parties, the adversary $\mathcal{A}$ and the simulator S. After a real or ideal execution, $\mathcal{Z}$ outputs a bit, which is considered as the output of the execution. The rationale behind this framework lies in showing that the environment $\mathcal{Z}$ (that represents all the things that happen outside of the protocol execution) is not able to efficiently distinguish between the real and ideal executions, thus implying that the real world protocol is as secure as the ideal functionality.

We denote by $\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z, \bar{r})$ the output of the environment $\mathcal{Z}$ in the real-world execution of protocol π between n parties with an adversary $\mathcal{A}$ under security parameter κ, input z and randomness $\bar{r} = (r_{\mathcal{Z}}, r_{\mathcal{A}}, r_{P_1}, \ldots, r_{P_n})$, where $(z, r_{\mathcal{Z}})$, $r_{\mathcal{A}}$ and r_{P_i} are respectively related to $\mathcal{Z}$, $\mathcal{A}$ and party i. Analogously, we denote by $\mathsf{IDEAL}_{\mathcal{F},\mathsf{S},\mathcal{Z}}(\kappa, z, \bar{r})$ the output of the environment in the ideal interaction between the simulator S and the ideal functionality $\mathcal{F}$ under security parameter κ, input z and randomness $\bar{r} = (r_{\mathcal{Z}}, r_{\mathsf{S}}, r_{\mathcal{F}})$, where $(z, r_{\mathcal{Z}})$, r_{S} and $r_{\mathcal{F}}$ are respectively related to $\mathcal{Z}$, S and $\mathcal{F}$. The real world execution is represented by the ensemble $\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{Z}} = \mathsf{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z, \bar{r}), \kappa \in \mathbb{N}, z \in \{0,1\}^*$ with a uniformly chosen $\bar{r}$, while the ideal execution is represented by $\mathsf{IDEAL}_{\mathcal{F},\mathsf{S},\mathcal{Z}} = \mathsf{IDEAL}_{\mathcal{F},\mathsf{S},\mathcal{Z}}(\kappa, z, \bar{r}), \kappa \in \mathbb{N}, z \in \{0,1\}^*$ with a uniformly chosen $\bar{r}$.

Our protocol is run in a $\mathcal{F}_{OT}$-hybrid model, *i.e.* we assume that the parties have access to a 1-out-of-2 OT ideal functionality $\mathcal{F}_{OT}$. In this model, honest parties do not communicate with the ideal functionality directly, but instead the adversary delivers all the messages to and from the ideal functionality. We consider the communication channels to be ideally authenticated, so that the adversary may read but not modify these messages. Unlike messages exchanged between parties, which can be read by the adversary, the messages exchanged between parties and the ideal functionality are divided into a *public header* and a *private header*. The public header can be read by the adversary and contains non-sensitive information (such as session identifiers, type of message, sender and receiver). On the other hand, the private header cannot be read by the adversary and contains information such as the parties' private inputs. We denote the ensemble of environment outputs that represents the execution of a protocol π in a $\mathcal{G}$-hybrid model as $\mathsf{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}$ (defined analogously to $\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$). UC security is then formally defined as:

Definition 2. *An n-party ($n \in \mathbb{N}$) protocol π is said to UC-realize an ideal functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid model if, for every adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that, for every environment $\mathcal{Z}$, the following relation holds:*

$$\mathsf{IDEAL}_{\mathcal{F},\mathsf{S},\mathcal{Z}} \overset{c}{\approx} \mathsf{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}$$

Adversarial Model: In this work we consider security against static adversaries, *i.e.* all parties are corrupted before the execution of the protocol begins. We consider active adversaries who may deviate from the protocol in any arbitrary way.

Setup Assumption: It is known that UC commitments (as well as most "interesting" functionalities) cannot be obtained in the plain model [CF01]. In order to overcome this impossibility, UC protocols require a *setup assumption*, that basically models a resource that is made available to all parties before execution starts. The security of our protocol is proved in the OT-hybrid model (referred to as the $\mathcal{F}_{OT}$-hybrid model in [Can01,CLOS02]), where all parties are assumed to have access to an ideal 1-out-of-2 OT functionality.

In this work we UC realize a homomorphic string commitment scheme, which is modeled by $\mathcal{F}_{\mathrm{HCOM}}$, a (modified) version of the $\mathcal{F}_{\mathrm{MCOM}}$ functionality introduced in [CLOS02]. We denote by $\mathcal{F}_{\mathrm{HCOM}}$ an instance of the functionality that takes messages in $\{0,1\}^{\lambda}$ as input. $\mathcal{F}_{\mathrm{HCOM}}$ basically adds commands for homomorphic operations over commitments and an abort in the Commit Phase to $\mathcal{F}_{\text{MCOM}}$. The abort is necessary to deal with inconsistent commitments that could be sent by a corrupted party. $\mathcal{F}_{\text{HCOM}}$ is defined in Figure 1.

As mentioned before, our protocol is proven to UC-realize $\mathcal{F}_{\text{HCOM}}$ in the $\mathcal{F}_{OT}$-hybrid model. 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 define $\mathcal{F}_{OT}$ in Figure 8 and $\mathcal{F}_{OT}^{t,n}$ in Figure 2 following the syntax of [CLOS02]. Once again, $\mathcal{F}_{OT}(\lambda)$ denotes an instance 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.

Functionality $\mathcal{F}_{OT}$

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

- Upon receiving a message $(\mathsf{sender}, sid, ssid, \boldsymbol{x}_0, \boldsymbol{x}_1)$ from P_s, where each $\boldsymbol{x}_i \in \{0,1\}^{\lambda}$, store the tuple $(ssid, \boldsymbol{x}_0, \boldsymbol{x}_1)$ (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 $(\mathsf{receiver}, sid, ssid, c)$ from P_r, check if a tuple $(ssid, \boldsymbol{x}_0, \boldsymbol{x}_1)$ was recorded. If yes, send $(\mathsf{received}, sid, ssid, \boldsymbol{x}_c)$ to P_r and $(\mathsf{received}, sid, ssid)$ to P_s and halt. If not, send nothing to P_r (but continue running).

Fig. 8. Functionality $\mathcal{F}_{OT}(\lambda)$

B Proofs of results for VSS

Lemma 1 (completeness for π_{VSS}**)** *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* $b = 1, \dots, \ell$.

Proof. The first claim is trivial. For the second, note first that $\pi_\ell(\boldsymbol{f}^b) = \boldsymbol{s}^b$ by construction of F, and second that each honest P_j holds the share $\boldsymbol{g}_j[b] = \boldsymbol{m}_j \cdot \boldsymbol{f}^b$.

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

Proof. Assume without loss of generality that (at least) players in the set $D = \{P_1, \dots, P_r\}$ are honest. For $b = 1, \dots, \ell$, let $\boldsymbol{r}_{D,b}$ be the reconstruction vectors guaranteed by r-reconstruction of $\mathcal{S}$. Then we define

$$\boldsymbol{v}^b = \sum_{i=1}^{r} \boldsymbol{r}_{D,b}[i] \cdot \boldsymbol{h}^i$$

Note also that since all players accept, all honest players accept and therefore we have $\boldsymbol{m}_j \cdot \boldsymbol{h}^i = \boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top$ for all honest P_i, P_j. It follows that for all honest P_j we have

$$\boldsymbol{m}_j \cdot \boldsymbol{v}^b = \sum_{i=1}^{r} \boldsymbol{r}_{D,b}[i] \cdot \boldsymbol{m}_j \cdot \boldsymbol{h}^i = \sum_{i=1}^{r} \boldsymbol{r}_{D,b}[i] \cdot \boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top .$$

Now, $\boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top$ is a scalar and hence equal to its own transpose. Hence $\boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top = (\boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top)^\top = \boldsymbol{m}_i \cdot \boldsymbol{g}_j^\top$. Plugging into the above sum, we get

$$\boldsymbol{m}_j \cdot \boldsymbol{v}^b = \sum_{i=1}^{r} \boldsymbol{r}_{D,b}[i] \cdot \boldsymbol{m}_i \cdot \boldsymbol{g}_j^\top = \boldsymbol{r}_{D,b} \cdot M_D \cdot \boldsymbol{g}_j^\top = \boldsymbol{g}_j[b].$$

which is exactly the b-th share held by P_j. We can define $\boldsymbol{s}_b$ by $\boldsymbol{s}^b[a] = \boldsymbol{v}^b[a]$ for $a = 1, \dots, \ell$, and the lemma follows.

Lemma 3 (privacy for π_{VSS}) *Suppose the dealer in π_{VSS} is honest and let $A = \{P_{i_1}, \dots, P_{i_t}\}$ be a set of players corrupted by the adversary. Now, in case 1 suppose the dealer executes π_{VSS} with input $\boldsymbol{s}^1, \dots, \boldsymbol{s}^\ell$ and then opens $T(\boldsymbol{s}^1, \dots, \boldsymbol{s}^\ell)$. In case 2, he executes π_{VSS} with input $\widetilde{\boldsymbol{s}}^1, \dots, \widetilde{\boldsymbol{s}}^\ell$ and then opens $T(\widetilde{\boldsymbol{s}}^1, \dots, \widetilde{\boldsymbol{s}}^\ell)$. If $T(\boldsymbol{s}^1, \dots, \boldsymbol{s}^\ell) = T(\widetilde{\boldsymbol{s}}^1, \dots, \widetilde{\boldsymbol{s}}^\ell)$, then the views of the adversary in the two cases are identically distributed.*

Proof. Let $m = \ell - \ell'$. We can assume $m > 0$ since if $m = 0$, full information on the secrets is released and there is nothing to prove. Let the row vectors $\boldsymbol{e}_1, \dots, \boldsymbol{e}_m$ be a basis for the kernel of T. Then, since $T(\boldsymbol{s}^1, \dots, \boldsymbol{s}^\ell) = T(\widetilde{\boldsymbol{s}}^1, \dots, \widetilde{\boldsymbol{s}}^\ell)$, there exists a set of coefficients $\alpha_{i,j}$ for $i = 1, \dots, \ell, j = 1, \dots, m$ such that

$$(\boldsymbol{s}^1[i], \dots, \boldsymbol{s}^\ell[i]) + \sum_{j=1}^{m} \alpha_{i,j} \cdot \boldsymbol{e}_j = (\widetilde{\boldsymbol{s}}^1[i], \dots, \widetilde{\boldsymbol{s}}^\ell[i])$$

or equivalently

$$\boldsymbol{s}^b[i] + \sum_{j=1}^{m} \alpha_{i,j} \cdot \boldsymbol{e}_j[b] = \widetilde{\boldsymbol{s}}^b[i]$$

Recall that by the t-privacy property of $\mathcal{S}$, for any $u = 1, \dots \ell$, there exist vectors $\boldsymbol{w}^{A,u}$ such that $\pi_\ell(\boldsymbol{w}^{A,u})$ has 1 in position u and 0 elsewhere and $M_A \cdot \boldsymbol{w}^{A,u} = \boldsymbol{0}$. Moreover we can construct row vectors $\boldsymbol{w}_v$ for $v = 1, \dots, m$ such that $\pi_\ell(\boldsymbol{w}_v) = \boldsymbol{e}_v$ and $M_A \cdot \boldsymbol{w}_v^\top = \boldsymbol{0}$. This is easily done by letting $\boldsymbol{w}_v$ be an appropriate linear combination of the transposes of the $\boldsymbol{w}^{A,u}$'s, for $u = 1, \dots, \ell$. We then define $W_{v,u} = \boldsymbol{w}^{A,u} \cdot \boldsymbol{w}_v$. Note that this is a matrix (rather than a scalar). Now, let F be any matrix that might be used for the sharing in case 1, and define

$$\widetilde{F} = F + W, \text{ where } W = \sum_{u=1}^{\ell} \sum_{v=1}^{m} \alpha_{u,v} \cdot W_{v,c}$$

Let $\boldsymbol{f}^b[a]$ be the entry in row a and column b of F (which is $\boldsymbol{s}^b[a]$). We therefore get, by construction of the $W_{v,u}$'s, that for $1 \leq a, b \leq \ell$:

$$\begin{aligned}\widetilde{\boldsymbol{f}}^b[a] &= \boldsymbol{f}^b[a] + \sum_{u=1}^{\ell}\sum_{v=1}^{m} \alpha_{u,v} \cdot W_{v,u}[a,b] \\ &= \boldsymbol{s}^b[a] + \sum_{u=1}^{\ell}\sum_{v=1}^{m} \alpha_{u,v} \cdot \boldsymbol{w}^{A,u}[a]\boldsymbol{w}_v[b] \\ &= \boldsymbol{s}^b[a] + \sum_{v=1}^{m} \alpha_{a,v} \cdot \boldsymbol{w}_v[b] \\ &= \boldsymbol{s}^b[a] + \sum_{v=1}^{m} \alpha_{a,v} \cdot \boldsymbol{e}_v[b] \\ &= \widetilde{\boldsymbol{s}}^b[a]\end{aligned}$$

So we see that $\widetilde{F}$ is a matrix that could have been used in case 2. Further, since by construction the $W_{v,u}$'s satisfy $M_A \cdot W_{v,u} = W_{v,u} \cdot M_A^\top =$ all-0 matrix, it is easy to see that then also $\widetilde{\boldsymbol{f}}_{i_k} = \boldsymbol{f}_{i_k}$ and $\widetilde{\boldsymbol{g}}_{i_k} = \boldsymbol{g}_{i_k}$ for any player P_{i_k} in A. So this means the adversary's view of the π_{VSS} protocol is the same whether we use F or $\widetilde{F}$.

Consider now the information the adversary sees when $T(\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell)$, respectively $T(\widetilde{\boldsymbol{s}}^1, \ldots, \widetilde{\boldsymbol{s}}^\ell)$, is opened. In case 1, the dealer reveals $T(F)$ and in the other case $T(F + W) = T(F) + T(W)$. We claim that in fact $T(W)$ is an all-0 matrix so that in fact the same information is revealed in the two cases. To see this, note that we have

$$T(W) = T(\sum_{u=1}^{\ell}\sum_{v=1}^{m} \alpha_{u,v} \cdot W_{v,u}) = \sum_{u=1}^{\ell}\sum_{v=1}^{m} \alpha_{u,v} \cdot T(W_{v,u})$$

Recall that $W_{v,u} = \boldsymbol{w}^{A,u} \cdot \boldsymbol{w}_v$. This means that if $\boldsymbol{r}$ is any row of $W_{v,u}$, then $\pi_\ell(\boldsymbol{r})$ is a multiple of $\boldsymbol{e}_v$. Since $\boldsymbol{e}_v$ is in the kernel of T, we have that $T(W_{v,u})$ is an all-0 matrix, and hence so is $T(W)$.

In conclusion, adding W induces a 1-1 mapping between matrices used in case 1 and in case 2, and this mapping keeps the adversary's view constant. The lemma follows.

Remark 1. We may generalize the privacy result even further: the dealer might execute several instances of π_{VSS} and open a linear combination of values from more than one of these instances. This can be shown to be secure in the same sense as we just proved by a similar argument, and we leave the details to the reader.

Remark 2. If we take T as the trivial function that maps all inputs to 0, then lemma 3 implies the usual privacy property for VSS protocols. In this case W is defined as $W = \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} \left(\widetilde{\boldsymbol{s}}^j[i] - \boldsymbol{s}^j[i]\right) \cdot \left(\boldsymbol{w}^{A,i} \cdot (\boldsymbol{w}^{A,j})^\top\right)$ and it allows the dealer to execute twice the π_{VSS} protocol (first with input $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell$ and after with input $\widetilde{\boldsymbol{s}}^1, \ldots, \widetilde{\boldsymbol{s}}^\ell$) in such a way that for each player in A (a set of at most t corrupted players) his share of $\boldsymbol{s}^b$ is equal to his share of $\widetilde{\boldsymbol{s}}^b$ for $b = 1, \ldots, \ell$. Indeed if in case 1 the dealer chose the matrix F to share $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell$, then in case 2 he can use the matrix $\widetilde{F} = F + W$ to share $\widetilde{\boldsymbol{s}}^1, \ldots, \widetilde{\boldsymbol{s}}^\ell$. By definition of W we have that $\boldsymbol{m}_i \cdot F = \boldsymbol{m}_i \cdot \widetilde{F}$ and $F \cdot \boldsymbol{m}_i^\top = \widetilde{F} \cdot \boldsymbol{m}_i^\top$ for any P_i in A.

Lemma 4 (randomness of the share vectors in π_{VSS}) *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.*

Proof. Suppose that V and $\widetilde{V}$ are two different $d \times \ell$ matrices such that $\pi_\ell(V) = \pi_\ell(\widetilde{V})$ and $M_A \cdot V = M_A \cdot \widetilde{V} = \pi^\ell(G_A)$ and call respectively $\boldsymbol{v}^i$ and $\tilde{\boldsymbol{v}}^i$ their columns. Observe that $M_A \cdot (\tilde{\boldsymbol{v}}^i - \boldsymbol{v}^i) = \boldsymbol{0}$ for any $i = 1, \ldots, \ell$. Let H be the matrix defined by

$$H = \sum_{i=1}^{\ell} \left[(\tilde{\boldsymbol{v}}^i - \boldsymbol{v}^i) \cdot \left(\boldsymbol{w}^{A,i}\right)^\top \right]$$

where $\boldsymbol{w}^{A,i}$ is a sweeping vector as defined in Section 2. It is easy to see that $\pi_\ell(H)$ is the all-0 matrix and that the i-th column of H is equal to $\tilde{\boldsymbol{v}}^i - \boldsymbol{v}^i$ for any $i = 1, \ldots, \ell$. Now consider a $d \times d$ matrix F such that $\pi^\ell(F) = V$ and define $\widetilde{F} = F + H$. It is clear that $\pi^\ell(\widetilde{F}) = \widetilde{V}$ and F has the secrets $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^\ell$ in the left-top corner. So if we prove that the adversary's view of Protocol π_{VSS} is the same whether F or $\widetilde{F}$ is used in the first step, then the lemma follows. The information given to the players in the set A during the execution of the π_{VSS} protocol with F or with $\widetilde{F}$ can be written respectively as $M_A \cdot F$, $F \cdot M_A^\top$ and $M_A \cdot \widetilde{F}$, $\widetilde{F} \cdot M_A^\top$. Furthermore

$$M_A \cdot \widetilde{F} = M_A \cdot F + M_A \cdot H$$
$$\widetilde{F} \cdot M_A^\top = F \cdot M_A^\top + H \cdot M_A^\top$$

Since $M_A \cdot (\tilde{\boldsymbol{v}}^i - \boldsymbol{v}^i) = \boldsymbol{0}$ for any $i = 1, \ldots, \ell$, by the definition of H we have that $M_A \cdot H$ is an all-0 matrix. The same holds for $H \cdot M_A^\top$ since the sweeping vectors satisfy $M_A \cdot \boldsymbol{w}^{A,i} = \boldsymbol{0}$ for any $i = 1, \ldots, \ell$.

Lemma 5 (soundness for π_{VSS}^φ) *If the dealer in π_{VSS}^φ is corrupt, but no player rejects, then for any $b = 1, \ldots, \ell$ 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 $\varphi\left(\boldsymbol{s}^b\right) = \boldsymbol{s}^{\varphi,b}$.*

Proof. From lemma 2 applied to both the correlated executions of π_{VSS}, we know that for any $b = 1, \ldots, \ell$ 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 $\boldsymbol{s}^b$ and $\boldsymbol{v}^{\varphi,b}$ shares $\boldsymbol{s}^{\varphi,b}$. In particular, from the proof of lemma 2 we know that

$$\boldsymbol{v}^b = \sum_{i=1}^{r} \boldsymbol{r}_{D,b}[i] \cdot \boldsymbol{h}^i \text{ and } \boldsymbol{s}^b = \pi_\ell(\boldsymbol{v}^b)$$

$$\boldsymbol{v}^{\varphi,b} = \sum_{i=1}^{r} \boldsymbol{r}_{D,b}[i] \cdot \boldsymbol{h}^{\varphi,i} \text{ and } \boldsymbol{s}^{\varphi,b} = \pi_\ell(\boldsymbol{v}^{\varphi,b})$$

where we assume w.l.o.g. that the players in $D = \{P_1, \dots, P_r\}$ are honest. Thus the condition $\varphi\left(\boldsymbol{s}^b\right) = \boldsymbol{s}^{\varphi,b}$ follows from the similar condition given for the first ℓ components of the vectors $\boldsymbol{h}^i$ and $\boldsymbol{h}^{\varphi,i}$.

Lemma 6 (privacy for π^{φ}_{VSS}) *Suppose the dealer in π^{φ}_{VSS} is honest. Now, in case 1 suppose the dealer executes π^{φ}_{VSS} with input $\boldsymbol{s}^1, \dots, \boldsymbol{s}^\ell$ and in case 2, he executes π^{φ}_{VSS} 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.*

Proof. Throughout this proof let $\boldsymbol{e}^i$ be the column vector of $\mathbb{F}^\ell$ that has 1 in position i and 0 elsewhere. By the t-privacy property of $\mathcal{S}$, for $c = 1, \dots, \ell$, there exist vectors $\boldsymbol{w}^{A,c}$ in $\mathbb{F}^d$ such that $\pi_\ell(\boldsymbol{w}^{A,c}) = \boldsymbol{e}^c$ and $M_A \cdot \boldsymbol{w}^{A,c} = \boldsymbol{0}$. As in remark 2, we define

$$W = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \left(\widetilde{\boldsymbol{s}}^j[i] - \boldsymbol{s}^j[i]\right) \cdot \left(\boldsymbol{w}^{A,i} \cdot (\boldsymbol{w}^{A,j})^\top\right)$$

Since the b-th column of $\boldsymbol{w}^{A,i} \cdot (\boldsymbol{w}^{A,j})^\top$ is the vector $\boldsymbol{w}^{A,j}[b] \cdot \boldsymbol{w}^{A,i}$, then

$$\begin{aligned}\pi_\ell(\boldsymbol{w}^b) &= \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \left(\widetilde{\boldsymbol{s}}^j[i] - \boldsymbol{s}^j[i]\right) \cdot \boldsymbol{w}^{A,j}[b] \cdot \boldsymbol{e}^i = \\ &= \sum_{j=1}^{\ell} \boldsymbol{w}^{A,j}[b] \cdot \sum_{i=1}^{\ell} \left(\widetilde{\boldsymbol{s}}^j[i] - \boldsymbol{s}^j[i]\right) \cdot \boldsymbol{e}^i = \\ &= \sum_{j=1}^{\ell} \boldsymbol{w}^{A,j}[b] \cdot \left(\widetilde{\boldsymbol{s}}^j - \boldsymbol{s}^j\right)\end{aligned}$$

where $\boldsymbol{w}^b$ is the b-th column of W. In particular $\pi_\ell(\boldsymbol{w}^b) = \widetilde{\boldsymbol{s}}^b - \boldsymbol{s}^b$ for any $b \leq \ell$.

Moreover there exists a set of coefficients $\beta_{i,j}$, with $i = 1, \dots, \ell$ and $j = 1, \dots, \ell$ such that

$$\varphi(\boldsymbol{e}^i) = \sum_{j=1}^{\ell} \beta_{i,j} \cdot \boldsymbol{e}^j$$

Now define for $i = 1, \dots, \ell$ column vectors $\boldsymbol{v}^{A,i} = \sum_{j=1}^{\ell} \beta_{i,j} \cdot \boldsymbol{w}^{A,j}$, observing that $\pi_\ell(\boldsymbol{v}^{A,c}) = \varphi(\boldsymbol{e}^c)$ and $M_A \cdot \boldsymbol{v}^{A,c} = \boldsymbol{0}$. Finally take

$$W_\varphi = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \left(\widetilde{\boldsymbol{s}}^j[i] - \boldsymbol{s}^j[i]\right) \cdot \left(\boldsymbol{v}^{A,i} \cdot (\boldsymbol{w}^{A,j})^\top\right)$$

The b-th column of $\boldsymbol{v}^{A,i} \cdot (\boldsymbol{w}^{A,j})^\top$ is the vector $\boldsymbol{w}^{A,j}[b] \cdot \boldsymbol{v}^{A,i}$. So if $\boldsymbol{w}^{\varphi,b}$ is the b-th column of W_φ, then

$$\begin{aligned}
\pi_\ell(\boldsymbol{w}^{\varphi,b}) &= \sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\left(\widetilde{\boldsymbol{s}}^j[i] - \boldsymbol{s}^j[i]\right) \cdot \boldsymbol{w}^{A,j}[b] \cdot \varphi(\boldsymbol{e}^i) = \\
&= \sum_{j=1}^{\ell} \boldsymbol{w}^{A,j}[b] \cdot \sum_{i=1}^{\ell}\left(\widetilde{\boldsymbol{s}}^j[i] - \boldsymbol{s}^j[i]\right) \cdot \varphi(\boldsymbol{e}^i) = \\
&= \sum_{j=1}^{\ell} \boldsymbol{w}^{A,j}[b] \cdot \left(\varphi(\widetilde{\boldsymbol{s}}^j) - \varphi(\boldsymbol{s}^j)\right)
\end{aligned}$$

In particular $\pi_\ell(\boldsymbol{w}^{\varphi,b}) = \varphi\left(\pi_\ell(\boldsymbol{w}^b)\right)$ and $\pi_\ell(\boldsymbol{w}^{\varphi,b}) = \varphi(\widetilde{\boldsymbol{s}}^b) - \varphi(\boldsymbol{s}^b)$ for any $b \leq \ell$.

Let F and F_φ be any pair of matrices that might be used in case 1 and define $\widetilde{F} = F + W$ and $\widetilde{F}_\varphi = F_\varphi + W_\varphi$. From the properties already observed for the columns of W and W_φ, it follows that $\widetilde{F}$ and $\widetilde{F}_\varphi$ form a pair or matrices that could have been used in case 2. Moreover, by construction of W and W_φ, we have that

$$M_A \cdot W = M_A \cdot W_\varphi = W \cdot M_A^\top = W_\varphi \cdot M_A^\top = \text{ all-0 matrix}$$

Thus it holds that $M_A \cdot F = M_A \cdot \widetilde{F}$, $F \cdot M_A^\top = \widetilde{F} \cdot M_A^\top$, $M_A \cdot F_\varphi = M_A \cdot \widetilde{F}_\varphi$ and $F_\varphi \cdot M_A^\top = \widetilde{F}_\varphi \cdot M_A^\top$. This means that the adversary's view is the same in the two cases and hence the lemma is proved.

Lemma 7 (randomness of the share vectors in π_{VSS}^φ) *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 $\pi_\ell(V) = (\boldsymbol{s}^1, \dots, \boldsymbol{s}^\ell)$, $\pi_\ell(V_\varphi) = (\varphi(\boldsymbol{s}^1), \dots, \varphi(\boldsymbol{s}^\ell))$, $M_A \cdot V = \pi^\ell(G_A)$ and $M_A \cdot V_\varphi = \pi^\ell(G_{\varphi,A})$ are equally likely, even given the adversary's entire view.*

Proof. Suppose that (V, V_φ) and $(\widetilde{V}, \widetilde{V}_\varphi)$ are two different pairs that satisfy the hypothesis of the lemma and repeat the same construction seen in the proof of Lemma 4 for each components.

Lemma 8 (soundness for $\pi_{VSS}^{\mathbf{0}}$) *If the dealer in $\pi_{VSS}^{\mathbf{0}}$ is corrupt, but no player rejects, then there exist column vectors $\boldsymbol{v}^1, \dots, \boldsymbol{v}^\ell$ each of which shares $\mathbf{0} \in \mathbb{F}^\ell$.*

Proof. Assume w.l.o.g. that the players in $D = \{P_1, \dots, P_r\}$ are honest and define

$$\boldsymbol{v}^b = \sum_{i=1}^{r} \boldsymbol{r}_{D,b}[i] \cdot \begin{pmatrix} \mathbf{0} \\ \boldsymbol{h}^{\mathbf{0},i} \end{pmatrix}$$

where $\boldsymbol{r}_{D,b}$ are the reconstruction vectors defined in Section 2. In the same way as in the proof of lemma 2, it follows from the condition $\boldsymbol{m}_{\mathbf{0},j} \cdot \boldsymbol{h}^{\mathbf{0},i} = \boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top$ that $\boldsymbol{v}^b$ shares $\mathbf{0}$.

Lemma 9 (randomness of the share vectors in $\pi^{\mathbf{0}}_{VSS}$) *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.*

Proof. Let V and $\widetilde{V}$ be two $e \times \ell$ matrices that satisfy $M_{\mathbf{0},A} \cdot V = M_{\mathbf{0},A} \cdot \widetilde{V} = \pi^{\ell}(G_A)$ and call respectively $\boldsymbol{v}^i$ and $\tilde{\boldsymbol{v}}^i$ their columns. Observe that $M_{\mathbf{0},A} \cdot (\tilde{\boldsymbol{v}}^i - \boldsymbol{v}^i) = \mathbf{0}$ for any $i = 1, \ldots, \ell$. Let H be the matrix defined by

$$H = \sum_{i=1}^{\ell} \left[(\tilde{\boldsymbol{v}}^i - \boldsymbol{v}^i) \cdot \left(\boldsymbol{w}^{A,i}\right)^{\top} \right]$$

where $\boldsymbol{w}^{A,i}$ is a sweeping vector as defined in Section 2. Since $\pi_{\ell}(\boldsymbol{w}^{A,i})$ has 1 in position i and 0 elsewhere, the i-th column of H is equal to $\tilde{\boldsymbol{v}}^i - \boldsymbol{v}^i$ for any $i = 1, \ldots, \ell$. Now consider a $e \times d$ matrix R such that $\pi^{\ell}(R) = V$ and define $\widetilde{R} = R + H$. It is clear that $\pi^{\ell}(\widetilde{R}) = \widetilde{V}$, so if we prove that the adversary view of Protocol $\pi^{\mathbf{0}}_{VSS}$ is the same whether R or $\widetilde{R}$ is used in the first step, then the lemma follows. The information given to the set A during the execution of Protocol $\pi^{\mathbf{0}}_{VSS}$ with R or with $\widetilde{R}$ can be written respectively as $M_{\mathbf{0},A} \cdot R$, $R \cdot M_A^{\top}$ and $M_{\mathbf{0},A} \cdot \widetilde{R}$, $\widetilde{R} \cdot M_A^{\top}$. Furthermore

$$M_{\mathbf{0},A} \cdot \widetilde{R} = M_{\mathbf{0},A} \cdot R + M_{\mathbf{0},A} \cdot H$$

$$\widetilde{R} \cdot M_A^{\top} = R \cdot M_A^{\top} + H \cdot M_A^{\top}$$

Since $M_{\mathbf{0},A} \cdot (\tilde{\boldsymbol{v}}^i - \boldsymbol{v}^i) = \mathbf{0}$ for any $i = 1, \ldots, \ell$, by the definition of H we have that $M_{\mathbf{0},A} \cdot H$ is an all-0 matrix. The same holds for $H \cdot M_A^{\top}$ since the sweeping vectors satisfy $M_A \cdot \boldsymbol{w}^{A,i} = \mathbf{0}$ for any $i = 1, \ldots, \ell$.

C Proof of Security for $\boldsymbol{\pi_{HCOM}}$

We prove the security of π_{HCOM} by constructing a simulator (or ideal adversary) that runs an internal copy of the real world adversary $\mathcal{A}$ and acts as the $\mathcal{F}^{t,n}_{OT}$ functionality to extract the committed values in the case of a corrupted sender, and uses the properties of π_{VSS} to handle the case of a corrupted receiver. The cases where the parties are both honest are handled trivially.

We analyze the cases when only P_s is corrupted and when only P_r is corrupted separately.

Simulating communication with $\mathcal{Z}$: The simulator S delivers all messages exchanged between $\mathcal{Z}$ and $\mathcal{A}$ as if they were communicating directly.

Simulating the setup phase when P_s is corrupted and P_r is honest:

1. Acting as $\mathcal{F}_{OT}^{t,n}$, S receives $(\mathsf{sender}, sid, ssid, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. S also receives and stores
$(sid, ssid, ((\widetilde{\boldsymbol{h}}^1)^\top, \widetilde{\boldsymbol{g}}_1), \ldots, ((\widetilde{\boldsymbol{h}}^n)^\top, \widetilde{\boldsymbol{g}}_n))$.
2. S uniformly samples a set of t indexes $c_1, \ldots, c_t \leftarrow [1, n]$, stores $((\boldsymbol{h}^{c_i})^\top, \boldsymbol{g}_{c_i}) = ((\widetilde{\boldsymbol{h}}^{c_i})^\top, \widetilde{\boldsymbol{g}}_{c_i}) - G(\boldsymbol{x}_{c_i}), 1 \leq i \leq t$ and runs the same check the receiver does in the protocol. If the check succeeds S continues, otherwise it sends abort to $\mathcal{F}_{\text{HCOM}}$ and halts.
3. S now attempts to reconstruct the secrets P_s has defined. It constructs a set H of consistent VSS players as follows: Initially H contains all players. Now, if H contains a pair of inconsistent players i, j, i.e., where we have $\boldsymbol{m}_j \cdot \boldsymbol{h}^i \neq \boldsymbol{g}_j \cdot \boldsymbol{m}_i^\top$, then these players are deleted from H. We repeat until no further pairs can be deleted. If the size of H is larger than the reconstruction threshold r, then S uses the reconstruction procedure from the proof of Lemma 2 to compute a set of secrets $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_\ell$. Otherwise, it sends abort to $\mathcal{F}_{\text{HCOM}}$ and halt.

Simulating the commit phase when P_s is corrupted and P_r is honest:

1. S stores $(sid, ssid, \eta, \widetilde{\boldsymbol{m}})$, computes $\boldsymbol{m} = \widetilde{\boldsymbol{m}} - \boldsymbol{r}_\eta$ and sends $(\mathsf{commit}, sid, ssid, P_s, P_r, \boldsymbol{m})$ to $\mathcal{F}_{\text{HCOM}}$.

Simulating the reveal phase when P_s is corrupted and P_r is honest:

1. S receives $(sid, ssid, \boldsymbol{m}, \boldsymbol{f}^\eta)$, 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, S sends $(\mathsf{reveal}, sid, ssid)$ to $\mathcal{F}_{\text{HCOM}}$. Otherwise, it does nothing.

Lemma 10. ***Statistical Binding:*** *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. When P_s is corrupted and P_r is honest, the following relation holds for every static active adversary $\mathcal{A}$ and every environment $\mathcal{Z}$:*

$$\mathsf{IDEAL}_{\mathcal{F}_{\text{HCOM}}, \mathsf{S}, \mathcal{Z}} \overset{s}{\approx} \mathsf{HYBRID}_{\pi_{HCOM}, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{OT}^{t,n}}$$

Proof. **Setup Phase:** The simulator S acts exactly like a honest receiver P_r would, so the distribution of the messages exchanged with $\mathcal{A}$ is exactly the same as in π_{HCOM}. The only way something different could happen in the simulation is therefore if the check of P_r succeeds, but S has to abort because the reconstruction of the $\boldsymbol{r}_i$'s fails. However, we show that this only happens with negligible probability: let $0 < \epsilon < 1$ be a constant such that $n - \epsilon n \geq r$, recall that r is δn for some constant δ so we choose such that $1 - \epsilon > \delta$. If the set H has size at least r then reconstruction succeeds, but if not, then the complement of H has size at least ϵn, and we now show that in this case the check of P_r succeeds with negligible probability. Recall that the complement of H consists of pairs of inconsistent players, so we split this set in two halves A, B of size at least $n\epsilon/2$ where each player in A has a "brother" in B he is inconsistent with. We will assume that the t players we check are chosen independently at random, choosing always t players as in the protocol incurs a larger chance of choosing bigger sets which will only increase the chance of catching an inconsistency. Consider choosing the first $t/2$ players to check. We expect that $t/2 \cdot \epsilon/2$ of these will be in A, and by a Chernoff bound we will see at least $t\epsilon/4 - \mu t$ except with negligible probability, where we can choose $\mu < \epsilon/4$. Let A' be this set. Now, when we choose the last $t/2$ players to check we will see an inconsistency if we choose even one of the brothers of players in A'. Since we have $t(\epsilon/4 - \mu)$ brothers and t is $\Theta(n)$, there is constant probability of hitting an inconsistency each time we choose a player, so missing all $t/2$ times happens with negligible probability.

Commit Phase: In the commit phase, S just follows the protocol.

Reveal Phase: Once again, the simulator S acts exactly like a honest receiver P_r would, so the distribution of the messages exanged with $\mathcal{A}$ is exactly the same as in π_{HCOM}. The only way to distinguish is therefore if $\mathcal{F}_{\text{HCOM}}$ behaves in a way that is inconsistent with the simulation. This can only happen if the corrupted P_s opens successfully a value $\boldsymbol{m}'$ that is different from the one S sent to $\mathcal{F}_{\text{HCOM}}$. From the above argument for the set-up phase and Lemma 2 we can assume that the of shares $\boldsymbol{c}_\eta$ of $\boldsymbol{r}_\eta$ determined in the set-up are all consistent, except for a subset of size at most ϵn (otherwise P_r's check would fail). Now, to change a set of $n - \epsilon n$ consistent shares into one that determines a different secret one must change at least $n(1-\epsilon) - r$ shares (r is the reconstruction threshold). This means that if P_s claims a secret different from $\boldsymbol{m}$ he must show us a set of shares that differ from $\boldsymbol{c}_\eta$ in at least $n(1-\epsilon) - r = n(1-\epsilon-\delta)$ positions, and since $1-\epsilon-\delta$ is a positive constant, the check by P_r will spot one of these changes except with negligible probability.

The above argument only explicitly applies to one run of the VSS. Consider now the case where we do many runs. Since we do the OTs only once, all runs will have the same watch list. Consider two runs, indexed by I and J. Each run will have its own sets, H_I and H_j, of consistent parties. When we add two commitments from run number I or run number J and open such a commitment, we can only be guaranteed that the resulting sharing is consistent for the parties $H_I \cap H_J$. So, we need that the size of $H_I \cap H_J$ stays above r, or the adversary might be able to open a sum of commitments from different blocks to a different value from the sum of the values in the commitments being summed. However, if $H_I \cap H_J$ does not have size at least r, then the complement has size at least ϵn. Furthermore, each "party" in the complement is caught either in execution I or in execution J, as P_r is checking both executions the same way. Hence it does not matter in which execution the inconsistency is located, it contributes to detection with the same probability as in the above analysis, and it follows as above that then the complement of $H_I \cap H_J$ can have size at most ϵn and hence $H_I \cap H_J$ has size at least r.

Simulating the setup phase when P_s is honest and P_r is corrupted:

1. For $i = 1, \dots, n$, S uniformly samples a random string $\boldsymbol{x}_i \in \{0,1\}^{\ell_{PRG}}$. Acting as $\mathcal{F}_{OT}^{t,n}$, upon receiving $(\mathsf{receiver}, sid, ssid, c_1, \dots, c_u)$, S sends $(\mathsf{received}, sid, ssid, \boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ to $\mathcal{A}$.
2. S uniformly samples k random strings $\boldsymbol{r}_i \leftarrow \{0,1\}^{\ell}$, $i = 1, \dots, n$ and runs π_{VSS} using $\boldsymbol{r}_1, \dots, \boldsymbol{r}_k$ as input, constructing n strings $((\boldsymbol{h}^i)^\top, \boldsymbol{g}_i), i = 1, \dots, n$ of length $2(\ell+e)$ from the vectors generated by π_{VSS}. S computes $((\widetilde{\boldsymbol{h}}^i)^\top, \widetilde{\boldsymbol{g}}_i) = ((\boldsymbol{h}^i)^\top, \boldsymbol{g}_i) + G(\boldsymbol{x}_i)$ for $i \in \{c_1, \dots, c_t\}$, randomly samples $((\widetilde{\boldsymbol{h}}^i)^\top, \widetilde{\boldsymbol{g}}_i) \leftarrow \{0,1\}^{2(\ell+e)}$ for $i \in \{1, \dots, n\} \setminus \{c_1, \dots, c_t\}$ and sends $(sid, ssid, ((\widetilde{\boldsymbol{h}}^1)^\top, \widetilde{\boldsymbol{g}}_1), \dots, ((\widetilde{\boldsymbol{h}}^n)^\top, \widetilde{\boldsymbol{g}}_n))$ to P_r.

Simulating the commit phase when P_s is honest and P_r is corrupted:

1. Upon receiving a message $(\mathsf{receipt}, sid, ssid, P_s, P_r)$ from $\mathcal{F}_{\mathrm{HCOM}}$, S chooses an unused random string $\boldsymbol{r}_\eta$, samples a random message $\boldsymbol{m}' \leftarrow \{0,1\}^{\ell}$, computes $\widetilde{\boldsymbol{m}} = \boldsymbol{m}' + \boldsymbol{r}_\eta$ and sends $(sid, ssid, \eta, \widetilde{\boldsymbol{m}})$ to $\mathcal{A}$.

Simulating the reveal phase when P_s is honest and P_r is corrupted:

1. Upon receiving a message $(\mathsf{reveal}, sid, ssid, P_s, P_r, \boldsymbol{m})$ from $\mathcal{F}_{OT}^{t,n}$, S constructs an appropriate $\overline{\boldsymbol{f}}^\eta$ such that $\pi_\ell(\overline{\boldsymbol{f}}^\eta) = \boldsymbol{m}' + \boldsymbol{r}_\eta - \boldsymbol{m}$ and $M\overline{\boldsymbol{f}}^\eta$ yields $\overline{\boldsymbol{g}}_{c_j}[\eta] = \boldsymbol{g}_{c_j}[\eta]$ for $j = 1, \dots, t$. S sends $(sid, ssid, \boldsymbol{m}, \overline{f}^\eta)$ to $\mathcal{A}$. Notice that a share vector $\overline{\boldsymbol{f}}^\eta$ with these properties can be constructed following the procedures of Lemma 3 as noted in Remark 2.

Lemma 11. ***Computational Hiding:*** *Let $G : \{0,1\}^{\ell_{PRG}} \to \{0,1\}^{2(\ell+e)}$ 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. When P_s is honest and P_r is corrupted, the following relation holds for every static active adversary $\mathcal{A}$ and every environment $\mathcal{Z}$:*

$$\mathsf{IDEAL}_{\mathcal{F}_{\mathrm{HCOM}},\mathsf{S},\mathcal{Z}} \overset{c}{\approx} \mathsf{HYBRID}_{\pi_{HCOM},\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{OT}^{t,n}}$$

Proof. Before we proceed to analyze the simulator, let's define a hybrid game where the encrypted shares $((\widetilde{\boldsymbol{h}}^i)^\top, \widetilde{\boldsymbol{g}}_i) = ((\boldsymbol{h}^i)^\top, \boldsymbol{g}_i) + G(\boldsymbol{x}_i)$ for $i \in \{1, \dots, n\} \setminus \{c_1, \dots, c_t\}$ are replaced by uniformly random strings $((\overline{\boldsymbol{h}}^i)^\top, \overline{\boldsymbol{g}}_i) \leftarrow \{0,1\}^{2(\ell+e)}$. It's clear that S can do that since it knows the choice values $\{c_1, \dots, c_t\}$. Suppose that both the real world execution and the ideal world simulation run this modified version of π_{HCOM}. Notice that any environment $\mathcal{Z}$ that distinguishes this game from a real execution of π_{HCOM} actually distinguishes between a uniformly random string and the output of the PRG, breaking the pseudorandomness property of the PRG.

We show that if both simulation and the real world execution run this computationally indistinguishable variant, then they are perfectly indistinguishable, this is clearly sufficient to show the lemma.

Setup Phase: The simulator S runs the setup exactly like a honest sender would do in the hybrid, resulting in the same distribution.

Commit Phase: S deviates from the protocol by sampling a random message $\boldsymbol{m}' \leftarrow \{0,1\}^{\ell}$ and committing to it by computing $\widetilde{\boldsymbol{m}} = \boldsymbol{m}' + \boldsymbol{r}_\eta$ and sending $(sid, ssid, \eta, \widetilde{\boldsymbol{m}})$ to $\mathcal{A}$, instead of committing to the real message $\boldsymbol{m}$. Nevertheless, notice that the $\boldsymbol{m}'$ is perfectly hidden by the random pad $\boldsymbol{r}_\eta$, which is in turn perfectly hidden from $\mathcal{A}$ since he only has access to t shares and the secret sharing schemes guarantees that the joint distribution of at most t shares is completely independent from the secret.

Reveal Phase: Upon input $(\mathsf{reveal}, sid, ssid, P_s, P_r, \boldsymbol{m})$, S computes an alternative random pad $\boldsymbol{r}'$ that yields $\boldsymbol{m} = \widetilde{\boldsymbol{m}} + \boldsymbol{r}'$. S then constructs a vector $\overline{\boldsymbol{f}^\eta}$ such that $M\overline{\boldsymbol{f}^\eta}$ results a share vector of $\boldsymbol{r}'$ where the shares revealed in the setup phase appear in the same positions, *i.e.* $M\boldsymbol{f}^\eta = (\overline{\boldsymbol{g}}_1[\eta], \ldots, \overline{\boldsymbol{g}}_n[\eta])^\top$ and such that that $\boldsymbol{g}_j[\eta] = \overline{\boldsymbol{g}}_j[\eta]$ for $j \in \{c_1, \ldots, c_t\}$. This allows the opening values sent by S to have exactly the same distribution as real opening values and pass the validity tests even though the revealed message is different. Notice that such a share vector can be constructed using the procedures in Remark 2. S sends $(sid, ssid, \boldsymbol{f}^\eta)$ to $\mathcal{A}$.

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 $\mathcal{F}_{\mathrm{HCOM}}$ in the $\mathcal{F}_{OT}^{t,n}(\ell_{PRG})$-hybrid model in the presence of static active adversaries. Formally, there exists an ideal adversary S such that the following relation holds for every static active adversary $\mathcal{A}$ and every environment $\mathcal{Z}$:*

$$\mathsf{IDEAL}_{\mathcal{F}_{\mathrm{HCOM}},\mathsf{S},\mathcal{Z}} \overset{c}{\approx} \mathsf{HYBRID}^{\mathcal{F}_{OT}^{t,n}}_{\pi_{HCOM},\mathcal{A},\mathcal{Z}}$$

Proof. The case where both parties are honest is straightforward. Thus, the proof follows from Lemma 10 and Lemma 11.

D The relation between our scheme and IPS

In this section, we give some details on the extent to which our scheme can be seen as the result of applying the IPS compiler to our VSS. The compiler starts from a multiparty outer protocol that implements some target functionality with active security. Together with an inner protocol that only needs to be passively secure and assuming access to OT, it builds a two party protocol implementing the same target functionality. The outer protocol is assumed to be in the client/server model with a small number of clients who have inputs and get outputs and many servers who execute the computation. Our VSS can be seen as having one client that has input (the dealer) and one that gets output, when the secret is opened. Thus, if IPS can be applied with the VSS as outer protocol, we will obtain a commitment protocol.

However, this will not work for the basic version of the IPS compiler because it assumes that the outer protocol has no communication directly between the servers. Our VSS does not satisfy this. But IPS suggests two ways to get rid of this limitation:

The first one involves changing the compiler to use a more complicated inner protocol. Using this method, we would obtain a commitment protocol that is inherently interactive, in contrast to our construction.

The second method involves precompiling the outer protocol by routing all server-to-server messages via (some of) the clients. The idea is that when a server wants to send a message, it will secret share it among two special clients who then sends the shares to the receiver. It is shown in the IPS paper that if this is done using what they call non-malleable secret sharing, the resulting protocol is still secure, but now with a smaller corruption threshold. We can use this method by adding two clients to the VSS to do the rerouting, they have no inputs or outputs and at most one of them can be corrupted.

We can apply the basic IPS compiler to this modified protocol, and then because only one client has input, we can make the inner protocol be a dummy protocol. This removes the interaction and the result is a commitment protocol that is somewhat similar to ours, but more complicated because of the rerouting in the underlying VSS and less efficient by a constant factor because of the extra secret sharing involved and the degradation of the corruption threshold.

E Using the VSS for MPC

In this section we sketch how to use our VSS to construct an MPC protocol as claimed in Theorem 3.

Preliminaries. First we note that the VSS protocols we describe are not full-fledged VSSs as they abort if there are conflicts found. But this can handled efficiently using known techniques for dispute control (see [BTH06]) as long as we do enough VSS instances in parallel. This way that the efficiency of our VSS is preserved up to a constant factor and additive terms. So in the following we will assume that we have VSS available in the standard sense.

We will write $\mathcal{S}(\boldsymbol{a}, \boldsymbol{f}_a)$ for the set of shares of vector $\boldsymbol{a}$ generated using randomness $\boldsymbol{f}_a$, similarly for $\hat{\mathcal{S}}$, where $\mathcal{S}$ is a commitment friendly secret sharing scheme, and will assume that no more than t players can be corrupted, where t is the privacy threshold. As before, we will choose the block size ℓ for the VSS such that it is linear in n, the number of servers.

We will also need that $\mathcal{S}$ and $\hat{\mathcal{S}}$ allow for error correction: if one is given a set of shares $\mathcal{S}(\boldsymbol{a}, \boldsymbol{f}_a)$ where a small enough constant fraction of the shares have been modified, then one can efficiently reconstruct the entire set of shares. This also means that (perhaps by solving a set of linear equations) one can come up with a vector $\boldsymbol{w}$ such that $\mathcal{S}(\boldsymbol{a}, \boldsymbol{w})$ equals the set of shares you reconstructed. In general it may not have to be the case that $\boldsymbol{w} = \boldsymbol{f}_a$. Everything we do in the following will work even if this is not the case, but for readability we will assume that they are equal, i.e., we assume that $\boldsymbol{f}_a$ can be reconstructed. The secret sharing scheme from [CC06] can easily be modified to suit our requirements, basically because it works similarly to Shamir's scheme, but with polynomials replaced by rational functions on smooth projective irreducible curves. We will need to work over a constant size extension field of characteristic 2 (size 64 will enough), but this only gives a constant factor overhead.

In our protocol, we will need a supply of pairs of random share vectors among the servers of form

$$\mathcal{S}(\boldsymbol{r}, \boldsymbol{f}_r), \hat{\mathcal{S}}(\boldsymbol{r}, \hat{\boldsymbol{f}}_r),$$

where $\boldsymbol{r}$ is random and unknown to the adversary. We do this by letting each server deal such a pair and then apply the linear extractor from Theorem 3 of [DI14]. More specifically, the extractor is linear mapping $\phi : \mathbb{F}^n \mapsto \mathbb{F}^u$ for some u that is linear in n, and if the servers have produced $\mathcal{S}(\boldsymbol{r}_i, \boldsymbol{f}_{r_i}), \hat{\mathcal{S}}(\boldsymbol{r_i}, \hat{\boldsymbol{f}}_{r_i})$ for $i = 1 \dots n$, we can build u pairs of sharings $\mathcal{S}(\boldsymbol{s}_j, \boldsymbol{f}_{s_j}), \hat{\mathcal{S}}(\boldsymbol{s_j}, \hat{\boldsymbol{f}}_{s_j})$ for $j = 1 \dots n$. We will do this by having the servers apply ϕ to the n shares they have received in the previous step. One can now observe that if we let $\boldsymbol{r}_i[k]$ be the k'th entry in the vector $\boldsymbol{r}_i$, then for each k, we will have

$$(\boldsymbol{s}_1[k], \dots, \boldsymbol{s}_u[k]) = \phi(\boldsymbol{r}_1[k], \dots, \boldsymbol{r}_n[k]).$$

Now, the adversary knows $\boldsymbol{r}_i[k]$ when the i'th server is corrupt and otherwise the value in completely unknown. Therefore the $\boldsymbol{r}_i$'s form a bit fixing source from the adversary's point of view and hence by Theorem 3 from [DI14], the outputs from ϕ are completely unknown to the adversary.

To have a single server S produce $\mathcal{S}(\boldsymbol{r}, \boldsymbol{f}_r), \hat{\mathcal{S}}(\boldsymbol{r}, \hat{\boldsymbol{f}}_r)$, we first run the VSS to produce the two sets of shares (actually ℓ of them in one go). We need to make sure that the same vector is contained by both sets of shares in a pair. Let $\mathcal{S}(1^\ell, \boldsymbol{f}_1)$ be a default sharing of the all-1 vector with public and fixed randomness. Then note that

$$\mathcal{S}(\boldsymbol{r}, \boldsymbol{f}_r) * \mathcal{S}(1^\ell, \boldsymbol{f}_1) - \hat{\mathcal{S}}(\boldsymbol{r}, \hat{\boldsymbol{f}}_r) = \hat{\mathcal{S}}(0^\ell, \boldsymbol{f}_r \otimes \boldsymbol{f}_1 - \hat{\boldsymbol{f}}_r)$$

Consider that we have ℓ instances of such differences (because we ran two VSS protocols in the first place), and that S knows the vectors of form $\boldsymbol{f}_r \otimes \boldsymbol{f}_1 - \hat{\boldsymbol{f}}_r$. If S chooses some more column vectors that also start with ℓ 0's, then we have exactly the set-up for the $\pi^{\mathbf{0}}_{VSS}$ protocol. So now we run this with S as dealer and if we get accept, then we know that the pairs of share vectors S produced are well formed.

Note that this gives us a general way for a player to show that two sets of shares, under $\mathcal{S}$, respectively $\hat{\mathcal{S}}$, determine the same secret, provided he knows the underlying random vectors.

We also need a supply of share vectors of form $\mathcal{S}(\boldsymbol{r}, \boldsymbol{f}_r), \mathcal{S}(\varphi(\boldsymbol{r}), \boldsymbol{f'}_r)$ for a linear function φ, this is easy, we just have all servers execute π^{φ}_{VSS} and use the linear extractor in the same way as above.

The Protocol. The actual protocol follows the pattern of many similar protocols: the clients start by VSS-ing their inputs, and then we work our way through the circuit, where by the nature of the secret sharing, we will be dong ℓ gates in parallel throughout. The standard representation of any vector $\boldsymbol{a}$ occurring in the computation is of form $\mathcal{S}(\boldsymbol{a}, \boldsymbol{f}_a)$.

Addition gates are trivial: by linearity, we have for any two share vectors that

$$\mathcal{S}(\boldsymbol{a}, \boldsymbol{f}_a) + \mathcal{S}(\boldsymbol{b}, \boldsymbol{f}_b) = \mathcal{S}(\boldsymbol{a} + \boldsymbol{b}, \boldsymbol{f}_a + \boldsymbol{f}_b).$$

For a multiplication gate, players will do local multiplication so we get

$$\mathcal{S}(\boldsymbol{a}, \boldsymbol{f}_a) * \mathcal{S}(\boldsymbol{b}, \boldsymbol{f}_b) = \hat{\mathcal{S}}(\boldsymbol{a} * \boldsymbol{b}, \boldsymbol{f}_a \otimes \boldsymbol{f}_b).$$

We now need to convert this to a sharing under $\mathcal{S}$ of the same value. To do this we take the next unused pair $\mathcal{S}(\boldsymbol{r}, \boldsymbol{f}_r), \hat{\mathcal{S}}(\boldsymbol{r}, \hat{\boldsymbol{f}}_r)$ and compute by local addition

$$\hat{\mathcal{S}}(\boldsymbol{a} * \boldsymbol{b}, \boldsymbol{f}_a \otimes \boldsymbol{f}_b) + \hat{\mathcal{S}}(\boldsymbol{r}, \hat{\boldsymbol{f}}_r) = \hat{\mathcal{S}}(\boldsymbol{a} * \boldsymbol{b} + \boldsymbol{r}, \boldsymbol{f}_a \otimes \boldsymbol{f}_b + \hat{\boldsymbol{f}}_r)$$

Now all servers send their shares in this value to a single server P_i. He reconstructs $\boldsymbol{a} * \boldsymbol{b} + \boldsymbol{r}$ and $\boldsymbol{f}_a \otimes \boldsymbol{f}_b + \hat{\boldsymbol{f}}_r$ which is possible by assumption on $\mathcal{S}$. He then creates $\mathcal{S}(\boldsymbol{a} * \boldsymbol{b} + \boldsymbol{r}, \boldsymbol{u})$ and sends shares to the servers. Assuming P_i did this correctly, we can compute

$$\mathcal{S}(\boldsymbol{a} * \boldsymbol{b} + \boldsymbol{r}, \boldsymbol{u}) - \mathcal{S}(\boldsymbol{r}, \boldsymbol{f}_r) = \mathcal{S}(\boldsymbol{a} * \boldsymbol{b}, \boldsymbol{u} - \boldsymbol{f}_r)$$

which was the goal. To make sure that P_i acted correctly, we have him prove that $\hat{\mathcal{S}}(\boldsymbol{a} * \boldsymbol{b} + \boldsymbol{r}, \boldsymbol{f}_a \otimes \boldsymbol{f}_b + \hat{\boldsymbol{f}}_r)$ and $\mathcal{S}(\boldsymbol{a} * \boldsymbol{b} + \boldsymbol{r}, \boldsymbol{u})$ contain the same value, and this can be done using the same procedure we had a client do above, if we make sure that P_i is assigned to handle at least ℓ multiplications.

Once the computation is done, we have share vectors of the output that we open towards the client who is to receive.

Routing Bits Between Layers. In order to allow us to evaluate the circuit using only parallel computation, we need to route the bits correctly between layers of the circuit. We can do this by using the [DIK10] approach to transform the circuit, or by computing the rerouting directly if the circuit is regular. In any case, this will require us to take a sharing $\mathcal{S}(\boldsymbol{a}, \boldsymbol{f}_a)$ and compute $\mathcal{S}(\varphi(\boldsymbol{a}), \boldsymbol{f'}_a)$ for a linear function φ where we can assume that the same function is to be applied to at least ℓ blocks. This can be done by taking a pair $\mathcal{S}(\boldsymbol{r}, \boldsymbol{f}_r), \mathcal{S}(\varphi(\boldsymbol{r}), \boldsymbol{f'}_r)$, compute

$$\mathcal{S}(\boldsymbol{a}, \boldsymbol{f}_a) + \mathcal{S}(\boldsymbol{r}, \boldsymbol{f}_r) = \mathcal{S}(\boldsymbol{a} + \boldsymbol{r}, \boldsymbol{f}_a + \boldsymbol{f}_r)$$

and send all shares of this value to a single player P_i. He can reconstruct $\boldsymbol{a} + \boldsymbol{r}$ and $\boldsymbol{f}_a + \boldsymbol{f}_r$, compute $\varphi(\boldsymbol{a} + \boldsymbol{r})$ and make a sharing $\mathcal{S}(\varphi(\boldsymbol{a} + \boldsymbol{r}), \boldsymbol{v})$. Assuming P_i acted correctly we can compute by local addition

$$\mathcal{S}(\varphi(\boldsymbol{a} + \boldsymbol{r}), \boldsymbol{v}) - \mathcal{S}(\varphi(\boldsymbol{r}), \boldsymbol{f'}_r) = \mathcal{S}(\varphi(\boldsymbol{a}), \boldsymbol{v} - \boldsymbol{f'}_r).$$

To check that P_i did this correctly, we observe that we have sharings $\mathcal{S}(\boldsymbol{a} + \boldsymbol{r}, \boldsymbol{f}_a + \boldsymbol{f}_r)$ and $\mathcal{S}(\varphi(\boldsymbol{a} + \boldsymbol{r}), \boldsymbol{v})$ where P_i knows the involved randomness. This can be seen as part of the set-up of the π^{φ}_{VSS} protocol. So P_i completes the set-up and we execute π^{φ}_{VSS} with P_i as dealer. If this accepts, we know that P_i acted correctly.

This completes the sketch of the protocol. The proof of security is straightforward.

Efficiency By inspection, one sees that every operation we do only incurs communication of at most a linear number of bits per bit of data we compute on. In particular this is true because the openings in the multiplication and linear function subprotocols are handled by a single player: an opening to one player communicates n bits, but the sharing contains $\Theta(n)$ bits. Therefore this protocol satisfies the claims of Theorem 2.