On the Download Rate of Homomorphic Secret Sharing[*]

Ingerid Fosli[†] Yuval Ishai[‡] Victor I. Kolobov[‡] Mary Wootters[§]

May 30, 2022

Abstract

A homomorphic secret sharing (HSS) scheme is a secret sharing scheme that supports evaluating functions on shared secrets by means of a local mapping from input shares to output shares. We initiate the study of the *download rate* of HSS, namely, the achievable ratio between the length of the output shares and the output length when amortized over ℓ function evaluations. We obtain the following results.

- In the case of *linear* information-theoretic HSS schemes for degree-d multivariate polynomials, we characterize the optimal download rate in terms of the optimal minimal distance of a linear code with related parameters. We further show that for sufficiently large ℓ (polynomial in all problem parameters), the optimal rate can be realized using Shamir's scheme, even with secrets over $\mathbb{F}_2$.
- We present a general rate-amplification technique for HSS that improves the download rate at the cost of requiring more shares. As a corollary, we get high-rate variants of computationally secure HSS schemes and efficient private information retrieval protocols from the literature.
- We show that, in some cases, one can beat the best download rate of linear HSS by allowing *nonlinear* output reconstruction and $2^{-\Omega(\ell)}$ error probability.

[*]This is a full version of [FIKW22].

[†]Google. ifosli@gmail.com. Work done while at Stanford University.

[‡]Technion. {yuvali,tkolobov}@cs.technion.ac.il. Research partially supported by ERC Project NTSC (742754), ISF grant 2774/20, and BSF grant 2018393.

[§]Stanford University. marykw@stanford.edu. Research partially supported by NSF grants CCF-1844628 and CCF-1814629, and by a Sloan Research Fellowship.

Contents

1 Introduction

Homomorphic Secret Sharing (HSS) [Ben86, BGI16a, BGI+18] is a form of secret sharing that supports computation on the shared data by means of locally computing on the shares. HSS can be viewed as a distributed analogue of homomorphic encryption [RAD78, Gen09] that allows for better efficiency and weaker cryptographic assumptions, or even unconditional security.

More formally, a standard t-private (threshold) secret-sharing scheme randomly splits an input x into k shares $x^{(1)}, \ldots, x^{(k)}$, distributed among k servers, so that no t of the servers learn any information about the input. (Here we assume *information-theoretic* security by default, but we will later also consider computational security.) A secret-sharing scheme as above is an HSS for a function class $\mathcal{F}$ if it additionally allows computation of functions $f \in \mathcal{F}$ on top of the shares. More concretely, an HSS scheme Π consists of three algorithms, Share, Eval and Rec. Given m inputs $x_1, \ldots, x_m$, which we think of as originating from m distinct *input clients*, the randomized Share function independently splits each input x_i among k servers. Each server j computes Eval on its m input shares and a target function $f \in \mathcal{F}$, to obtain an *output share* $y^{(j)}$. These output shares are then sent to an *output client*, who runs $\mathsf{Rec}(y^{(1)}, \ldots, y^{(k)})$ to reconstruct $f(x_1, \ldots, x_m)$.[1]

As described up to this point, the HSS problem admits a trivial solution: simply let Eval be the identity function (which outputs all input shares along with the description of f) and let Rec first reconstruct the m inputs and then compute f. To be useful for applications, HSS schemes are required to be *compact*, in the sense that the output shares $y^{(j)}$ are substantially shorter than what is sent in this trivial solution. A strong compactness requirement, which is often used in HSS definitions from the literature, is additive reconstruction. Concretely, in an *additive HSS* scheme the output of f is assumed to come from an Abelian group $\mathbb{G}$, each output share $y^{(j)}$ is in $\mathbb{G}$, and Rec simply computes group addition. Simple additive HSS schemes for linear functions [Ben86], finite field multiplication [BGW88, CCD88, CDM00], and low-degree multivariate polynomials [BF90, BFKR90, CGKS98] are implicit in classical protocols for information-theoretic secure multiparty computation and private information retrieval. More recently, *computationally secure* additive HSS schemes were constructed for a variety of function classes under a variety of cryptographic assumptions [BGI15, DHRW16, BGI16a, BGI16b, FGJI17, BGI+18, BKS19, BCG+19, CM21, OSY21, RS21].

While additive HSS may seem to achieve the best level of compactness one could hope for, allowing for 1-bit output shares when evaluating a Boolean function f, it still leaves a factor-k gap between the output length and the total length of the output shares communicated to the output client. This is undesirable when f has a long output, especially when k is big. We refer to the total output share length of Π as its *download cost* and to the ratio between the output length and the download cost as its *download rate* or simply *rate*. We note that even when allowing a bigger number of servers and using a non-additive output encoding, it is not clear how to optimize the rate of existing HSS schemes (see Section 1.1.1 for further discussion).

In the related context of homomorphic encryption, it was recently shown that the download rate, amortized over a long output, can approach 1 at the limit [DGI+19, GH19, BDGM19]. However, here the concrete download cost must inherently be bigger than a cryptographic security parameter, and the good amortized rate only kicks in for big output lengths that depend polynomially on the

[1] One may also consider *robust* HSS in which reconstruction can tolerate errors or erasures. While some of our results can be extended to this setting, in this work (as in most of the HSS literature) we only consider the simpler case of non-robust reconstruction.

security parameter. The relaxed HSS setting has the qualitative advantage of allowing the rate to be independent of any security parameters, in addition to allowing for information-theoretic security and better concrete efficiency.

In this work, we initiate the systematic study of the download cost of homomorphic secret sharing. We ask the following question:

> *How compact can HSS be? Can existing HSS schemes be modified to achieve amortized download rate arbitrary close to* 1*, possibly by employing more servers?*

More concretely, our primary goal is to understand the best download rate attainable given the number of servers k, the security threshold t and the class of functions $\mathcal{F}$ that the HSS is guaranteed to work for. As a secondary goal, we would also like to minimize the overhead to the *upload cost*, namely the total length of the input shares.

To help establish tight bounds, we study the download rate when amortized over multiple instances. That is, given inputs $x_{i,j}$ for $i \in [m], j \in [\ell]$, all shared separately, and functions $f_1, \ldots, f_\ell \in \mathcal{F}$, we consider the problem of computing $f_j(x_{1,j}, x_{2,j}, \ldots, x_{m,j})$ for all $j \in [\ell]$. (Note that positive results in this setting also apply in the easier settings of computing multiple functions on the same inputs or the same function on multiple inputs.) HSS with a big number of instances ℓ can arise in many application scenarios that involve large-scale computations on secret-shared inputs. This includes private information retrieval, private set intersection, private statistical queries, and more. Such applications will motivate specific classes $\mathcal{F}$ we consider in this work.

1.1 Contributions

We develop a framework in which to study the download rate of HSS and obtain both positive and negative results for special cases of interest. In more detail, we make the following contributions.

1.1.1 Optimal-download linear HSS for low-degree polynomials, and applications

We consider information-theoretic HSS when the function class $\mathcal{F}$ is the set of degree-d m-variate polynomials over a finite field $\mathbb{F}$. A standard HSS for this class [BF90, BFKR90] uses $k = dt + 1$ servers and has download rate of $1/k$. By using $k \gg dt$ servers, a multi-secret extension of Shamir's secret sharing scheme [FY92] can be used to get the rate arbitrarily close to $1/d$, for sufficiently large ℓ.[2] In Section 3 we give two constructions that obtain a better rate, arbitrarily close to 1.

Our first construction (Theorem 3) is based on the highly redundant *CNF sharing* [ISN89], where each input is shared by replicating $\binom{k}{t}$ additive shares. This construction is defined for all choices of $\mathbb{F}, m, d, t, \ell$ and $k > dt$, and its rate is determined by the best minimal distance of a linear code with related parameters. For sufficiently large ℓ this code is an MDS code, in which case the rate is $1 - dt/k$. The main downside of this construction is a $\binom{k}{t}$ overhead to the upload cost. Settling for computational security, this overhead can be converted into a *computational* overhead (which is reasonable in practice for small values of k, t) by using a pseudorandom secret sharing technique [GI99, CDI05].

Our second construction (Theorem 5) uses *Shamir sharing* [Sha79], where each input is shared by evaluating a random degree-t polynomial over an extension field of $\mathbb{F}$ at k distinct points. This

[2] Intuitively, this is because one can use polynomials of degree $\approx k/d$ to share the secrets (yielding rate $\approx 1/d$ when $k \gg dt$). A higher degree is not possible because the product of d polynomials should have degree $< k$ to allow interpolation. See Remark 4 for more details.

construction also achieves a download rate of $1 - dt/k$ for sufficiently large ℓ, but here this rate is achieved with upload cost that scales polynomially with t and the other parameters.

Both constructions are *linear* in the sense that Share and Rec are linear functions. In Section 3.2 we show that for such linear HSS schemes, $1 - dt/k$ is the best rate possible, implying optimality of our schemes.

We compare the above two HSS schemes in Section 3.1.3. Briefly, the Shamir-based scheme has better upload cost (which scales polynomially with all parameters) but is more restrictive in its paramater regime: that is, it only yields an optimal scheme in a strict subset of the parameter settings where the CNF-based scheme is optimal. One may wonder if this is a limitation of our Shamir-based scheme in particular or a limitation of Shamir sharing in general. We show in Proposition 1 that it is the latter. That is, there are some parameter regimes where *no* HSS based on Shamir sharing can perform as well as an HSS based on CNF sharing.

Applications: High-rate PIR and more. We apply our HSS for low-degree polynomials to obtain the first information-theoretic private information retrieval (PIR) protocols that simultaneously achieve low (sublinear) upload cost and near-optimal download rate that gets arbitrarily close to 1 when the number of servers grows. A t-private k-server PIR protocol [CGKS98] allows a client to retrieve a single symbol from a database in $\mathcal{Y}^N$, which is replicated among the servers, such that no t servers learn the identity of the retrieved symbol. The typical goal in the PIR literature is to minimize the communication complexity when $\mathcal{Y} = \{0, 1\}$. In particular, the communication complexity should be sublinear in N. Here we consider the case where the database has (long) ℓ-bit records, namely $\mathcal{Y} = \{0, 1\}^\ell$. Our goal is to maximize the download rate while keeping the upload cost sublinear in N. Chor et al. [CGKS98] obtain, for any integers $d, t \geq 1$ and $k = dt + 1$, a t-private k-server PIR protocol with upload cost $O(N^{1/d})$ and download rate $1/k$ (for sufficiently large ℓ). This protocol implicitly relies on a simple HSS for degree-d polynomials. Using our high-rate HSS for degree-d polynomials, by increasing the number of servers k the download rate can be improved to $1 - dt/k$ (in particular, approach 1 when $k \gg dt + 1$) while maintaining the same asymptotic upload cost. See Theorem 6 for a formal statement.

It is instructive to compare this application to a recent line of work on the download rate of PIR. Sun and Jafar [SJ16, SJ17, SJ18], following [SRR14], have shown that the optimal download rate of t-private k-server PIR is $(1 - t/k)/(1 - (t/k)^N)$. However, their positive result has $\Omega(N)$ upload cost. We get a slightly worse[3] download rate of $1 - dt/k$, where the upload cost is sublinear for $d \geq 2$.

Finally, beyond PIR, HSS for low-degree polynomials can be directly motivated by a variety of other applications. For instance, an inner product between two integer-valued vectors (a degree-2 function) is a measure of correlation. To amortize the download rate of computing ℓ such correlations, our HSS scheme for degree-2 polynomials over a big field $\mathbb{F}$ can be applied. As another example, the intersection of d sets $S_i \subseteq [\ell]$, each represented by a characteristic vector in $\mathbb{F}_2^\ell$, can be computed by ℓ instances of a degree-d monomial over $\mathbb{F}_2$. See [LMS18, ILM21] for more examples.

[3]Note that our positive result applies also to a stronger variant of amortized PIR, which amortizes over ℓ *independent* instances of PIR with databases in $\{0, 1\}^N$. In this setting, our construction with $d = 1$ achieves an optimal rate of $1 - t/k$ (where optimality follows from [SRR14] or from Lemma 3). In Section 1.1.2 below we discuss a construction of *computationally* secure PIR that achieves the same rate of $1 - 1/k$ (for $t = 1$) with *logarithmic* upload cost.

1.1.2 Black-box rate amplification for additive HSS

The results discussed so far are focused on information-theoretic HSS for a specific function class. Towards obtaining other kinds of high-rate HSS schemes, in Section 4 we develop a general black-box transformation technique (Lemma 4) that can improve the download rate of any *additive* HSS (where Rec adds up the output shares) by using additional servers. More concretely, the transformation can obtain a high-rate t-private k-server HSS scheme Π by making a *black-box* use of any additive t_0-private k_0-server Π_0, for suitable choices of k_0 and t_0. The transformation typically has a small impact on the upload cost and applies to both information-theoretically secure and computationally secure HSS. While we cannot match the parameters of the HSS for low-degree polynomials (described above) by using this approach, we can apply it to other function classes and obtain rate that approaches 1 as k grows.

We present three useful instances of this technique. In the first (Theorem 8), we use Π_0 with $k_0 = \binom{k}{t}$ and $t_0 = k_0 - 1$ to obtain a t-private k-server HSS Π with rate $1 - t/k$. Combined with a computational HSS for circuits from [DHRW16] (which is based on a variant of the Learning With Errors assumption) this gives a general-purpose computationally t-private HSS with rate $1 - t/k$, approaching 1 when $k \gg t$, at the price of upload cost and computational complexity that scale with $\binom{k}{t}$. This should be compared to the $1/(t+1)$ rate obtained via a direct use of [DHRW16]. Note that, unlike recent constructions of "rate-1" fully homomorphic encryption schemes [GH19, BDGM19], here the concrete download rate is independent of the security parameter.

The above transformation is limited in that it requires Π_0 to have a high threshold t_0, whereas most computationally secure HSS schemes from the literature are only 1-private. Our second instance of a black-box transformation (Theorem 9) uses any 1-private 2-server Π_0 to obtain a 1-private k-server Π with rate $1 - 1/k$. Applying this to HSS schemes from [BGI15, BGI16b], we get (concretely efficient) 1-private k-server computational PIR schemes with download rate $1 - 1/k$, based on any pseudorandom generator, with upload cost $O(\lambda k \log(N))$ (where λ is a security parameter). By using a different approach, the authors in [HH19] show that when k is a power of two an upload cost of $O(\lambda \log(k) \log(N))$ is sufficient. We can also apply this transformation to 1-private 2-server HSS schemes from [BGI16a, OSY21, RS21], obtaining 1-private k-server HSS schemes for branching programs based on number-theoretic cryptographic assumptions (concretely, DDH or DCR), with rate $1 - 1/k$.

Our third and final instance of the black-box transformation is motivated by the goal of *information-theoretic* PIR with sub-polynomial ($N^{o(1)}$) upload cost and download rate approaching 1. Here the starting point is a 1-private 3-server PIR scheme with sub-polynomial upload cost based on matching vectors [Yek07, Efr09]. While this scheme is not additive, it can be made additive by doubling the number of servers. We then (Theorem 11) apply the third variant of the transformation to the resulting 1-private 6-server PIR scheme, obtaining a 1-private k-server PIR scheme with sub-polynomial upload cost and rate $1 - 1/\Theta(\sqrt{k})$. Note that here we cannot apply the previous transformation since $k_0 = 6 > 2$.

We leave open the question of fully characterizing the parameters for which such black-box transformations exist.

1.1.3 Nonlinear download rate amplification

All of the high-rate HSS schemes considered up to this point have a *linear* reconstruction function Rec. Moreover, they all improve the rate of existing baseline schemes by increasing the number of servers. In Section 5 we study the possibility of circumventing this barrier by relaxing the linearity requirement, without increasing the number of servers.

The starting point is the following simple example. Consider the class $\mathcal{F}$ of degree-d monomials (products of d variables) over a field $\mathbb{F}$ of size $|\mathbb{F}| \approx d$. Letting $k = d + 1$ and $t = 1$, we have the following standard "baseline" HSS scheme: Share applies Shamir's scheme (with $t = 1$) to each input; to evaluate a monomial f, Eval (computed locally by each server) multiplies the shares of the d variables of f; finally Rec can recover the output by interpolating a degree-d polynomial, applying a linear function to the output shares. The key observation is that since each input share is uniformly random over $\mathbb{F}$, the output of Eval is biased towards 0. Concretely, by the choice of parameters, each output share is 0 except with $\approx 1/e$ probability. It follows that when amortizing over ℓ instances, and settling for $2^{-\Omega(\ell)}$ failure probability, the output of Eval can be compressed by roughly a factor of e by simply listing all $\approx \ell/e$ nonzero entries and their locations.

While this example already circumvents our negative result for linear HSS, it only applies to evaluating products over a big finite field, which is not useful for any applications we are aware of. Moreover, this naive compression method does not take advantage of correlations between output shares. In Theorem 12 we generalize and improve this method by using a variant of Slepian-Wolf coding tailored to the HSS setting. Note that we cannot use the Slepian-Wolf theorem directly, because the underlying joint distribution depends on the output of f and is thus not known to each server. We apply our general methodology to the simple but useful case where f computes the AND of two input bits. As discussed above, ℓ instances of such f can be motivated by a variant of the private set intersection problem in which the output client should learn the intersection of two subsets of $[\ell]$ whose characteristic vectors are secret-shared between the servers. By applying Theorem 12 to a 1-private 3-server HSS for AND based on CNF sharing, we show (Corollary 6) that the download rate can be improved from 1/3 to ≈ 0.376 (with $2^{-\Omega(\ell)}$ failure probability), which is the best possible using our general compression method.

Perhaps even more surprisingly, the improved rate can be achieved while ensuring that the output shares reveal no additional information except the output (Proposition 4). We refer to the latter feature as *symmetric privacy.* This should be contrasted with the above example of computing a monomial over a large field, where the output shares *do* reveal more than just the output (as they reveal the product of d degree-1 polynomials that encode the inputs). While symmetric privacy can be achieved by rerandomizing the output shares—a common technique in protocols for secure multiparty computation [BGW88, CCD88, CDM00]—this eliminates the possibility for compression.

Our understanding of the rate of nonlinear HSS is far from being complete. Even for simple cases such as the AND function, some basic questions remain. Does the compression method of Theorem 12 yield an optimal rate? Can the failure probability be eliminated? Can symmetric privacy be achieved with nontrivial rate even when the output client may collude with an input client? We leave a more systematic study of these questions to future work.

1.2 Technical Overview

In this section we give a high level overview of the main ideas used by our results.

1.2.1 Linear HSS for low-degree polynomials

We begin by describing our results in Section 3 for linear HSS. We give positive and negative results; we start with the positive results.

HSS for Concatenation. Both of our constructions of linear HSS (the first based on CNF sharing, the second on Shamir sharing) begin with a solution for the special problem of *HSS for concatenation* (Definition 10). Given ℓ inputs $x_1, \ldots, x_\ell$ that are shared separately, the goal is for the servers to produce output shares (the outputs of Eval) that allow for the joint recovery of $(x_1, \ldots, x_\ell)$, while still using small communication. Once we have an HSS for concatenation based on either CNF sharing (Definition 6) or Shamir sharing (Definition 7), an HSS for low-degree polynomials readily follows by exploiting the specific structure of CNF or Shamir.[4]

This problem can be viewed as an instance of *share conversion.* Concretely, the problem is to locally convert from a linear secret sharing scheme (LSSS) that shares $x_1, \ldots, x_\ell$ *separately* (via either CNF or Shamir) to a linear multi-secret sharing scheme (LMSSS) that shares $x_1, \ldots, x_\ell$ *jointly.* Thus, our constructions follow by understanding such share conversions.

Construction from CNF sharing. If we begin with CNF sharing, we can completely characterize the best possible share conversions as described above. Because t-private CNF shares can be locally converted to *any* $\geq t$-private LMSSS (Corollary 1, extending [CDI05] from LSSS to LMSSS), the above problem of share conversion collapses to the problem of understanding the best rate attainable by an LMSSS with given parameters. It is well-known that LMSSS's can be constructed from linear error correcting codes with good dual distance (see, e.g., [Mas95, CCG$^+$07]). However, in order to construct HSS we are interested only in t-private LMSSS with the property that *all k* parties can reconstruct the secret (as opposed to any $t+1$ parties or some more complicated access structure), which results in a particularly simple correspondence (Lemma 1, generalizing [GM10]). This in turn leads to Theorem 2, which *characterizes* the best possible download rate for any linear HSS-for-concatenation in terms of the best trade-off between the rate and distance of a linear code. This theorem gives a characterization (a negative as well as a positive result). The positive result (when we plug in good codes) gives our CNF-based HSS-for-concatenation, which leads to Theorem 3, our CNF-based HSS for general low-degree polynomials.

Construction from Shamir sharing. If we begin with Shamir sharing, we can no longer locally convert to any LMSSS we wish. Instead, we develop a local conversion to a specific LMSSS with good rate. In order to develop this construction, we leverage ideas from the *regenerating codes* literature (see the discussion in Section 1.3 below). Unfortunately, we are not able to use an off-the-shelf regenerating code for our purposes, but instead we take advantage of some differences between the HSS setting and the regenerating code setting in order to construct a scheme that suits our needs. This results in Theorem 4 for HSS-for-concatenation, and then Theorem 5 for HSS for general low-degree polynomials.

As an application of our Shamir-based construction, we extend information-theoretic PIR protocols from [CGKS98] to allow better download rate by employing more servers, while maintaining the same (sublinear) upload cost. This leads to Theorem 6.

[4]In some parameter regimes, HSS for concatenation with optimal download rate is quite easy to achieve using other secret sharing schemes, such as the multi-secret extension of Shamir's scheme due to Franklin and Yung [FY92]. However, this does not suffice for obtaining rate-optimal solutions for polynomials of degree $d > 1$ (see Remark 4).

Negative results. As mentioned above, Theorem 2 contains both positive and negative results, with the negative results stemming from negative results about the best possible trade-offs between the rate and distance of linear codes. This shows that our CNF-based construction is optimal for HSS-for-concatenation, but unfortunately does not extend to give a characterization of the best download rate for HSS for low-degree polynomials. Instead, in Theorem 7, we use a linear-algebraic argument to show that no linear HSS for degree d polynomials can have download rate better than $1-dt/k$. This means that for sufficiently large ℓ, both of our HSS schemes for low-degree polynomials have an optimal rate.

1.2.2 Black-box rate amplification for additive HSS

Next, we give a brief technical overview of our results in Section 4 on black-box rate amplification for additive HSS.

The general approach. We show that starting from any t_0-private k_0-server HSS scheme Π_0 with additive reconstruction (over some finite field), it is possible to construct other t-private k-server HSS schemes with higher rate. The main observation is that due to the additive reconstruction property, after the servers perform their evaluation, the output shares form an additive sharing of the output $y = y^{(1)} + \ldots + y^{(k_0)}$ (which is t_0-private). By controlling how the shares are replicated among the servers, each output y_i, $i = 1, \ldots, \ell$, is shared among the servers according to some LSSS. Hence, at this stage, this becomes a share conversion problem, where we want to convert separately shared outputs into a high-rate joint LMSSS, which yields our high-rate HSS scheme.

Black-box transformations with large k_0. We observe that if $k_0 = \binom{k}{t}$ and $t_0 = k_0 - 1$, then we can replicate the shares of Π_0 in such a manner that each output y is t-CNF shared among the servers. Concretely, if $y = y^{(1)} + \ldots + y^{(k_0)}$, then we can identify each index $i = 1, \ldots, k_0$ with a subset $T_i \in \binom{[k]}{t}$, and provide each server $j = 1, \ldots, k$ with $y^{(i)}$ if and only if $j \notin T_i$, after which the servers hold a t-CNF sharing of y (Definition 6). Therefore, as in Section 1.1.1, this now reduces to finding a t-private LMSSS with the best possible rate.

Black-box transformations with $k_0 = 2$. Most computationally secure HSS schemes from the literature are 1-private 2-server schemes, to which the previous transformation does not apply. Our second transformation converts any (additive) 1-private 2-server Π_0 to a 1-private k-server Π with rate $1 - 1/k$. This is obtained by replicating $k - 1$ pairs of (output) shares among the k servers in a way that: (1) each server gets only one share from each pair; (2) the servers can locally convert their shares to a 1-private $(k-1)$-LMSSS of the outputs of rate $1-1/k$. To illustrate this approach for $k = 3$, suppose we are given a 2-additive secret sharing for every output $y_i = y_i^{(1)} + y_i^{(2)}$, $i = 1, 2$. We obtain a 1-private 3-server 2-LMSSS sharing of the outputs with information rate $2/3$ in the following way:

$$\begin{tikzcd}
 & y_1^{(1)}, y_2^{(1)} \arrow[r, "y_1^{(1)} + y_2^{(1)}"] & z^{(1)} \arrow[dr] \arrow[dddr] & \\
y_1 \arrow[ur] \arrow[dr] \arrow[dddr] & & & z^{(1)} + z^{(2)} = y_1 \\
 & y_1^{(2)}, y_2^{(1)} \arrow[r, "y_1^{(2)} - y_2^{(1)}"] & z^{(2)} \arrow[ur] & \\
y_2 \arrow[uuur] \arrow[ur] \arrow[dr] & & & z^{(1)} + z^{(3)} = y_2 \\
 & y_1^{(1)}, y_2^{(2)} \arrow[r, "-y_1^{(1)} + y_2^{(2)}"] & z^{(3)} \arrow[ur] &
\end{tikzcd}$$

Since we need only 3 shares to reconstruct 2 secrets, the rate is 2/3. This can be generalized in a natural way to k servers and $k-1$ outputs.

Sub-polynomial upload cost PIR with high rate. Our third variant of the black-box transformation is motivated by the goal of high-rate information-theoretic PIR with sub-polynomial upload cost. Unlike the previous parts, here we start with a 1-private 6-server HSS, which has a lower privacy-to-servers ratio. While we don't obtain a tight characterization for this parameter setting, we can reduce the problem to a combinatorial packing problem, which suffices to get rate approaching 1. Concretely, for a universe $\{1, \ldots, q\}$ we need the largest possible family of subsets $\mathcal{S} \subseteq \binom{[q]}{5}$, such that distinct sets in $\mathcal{S}$ have at most a single element in common. Next, we show that it is possible to associate every set in $\mathcal{S}$ with a secret from $\mathbb{F}$, and also every set in $\mathcal{S}$ *and* an element of the universe $\{1, \ldots, q\}$ with a server, in such a way that the output shares, each an element of $\mathbb{F}$, constitute a high-rate LMSSS sharing of the outputs. This gives us a rate of $1 - q/(q + |\mathcal{S}|)$. Using known constructions of subset families as above of size $\Theta(q^2)$ [EH63], we get a download rate of $1 - 1/\Theta(q) = 1 - 1/\Theta(\sqrt{k})$.

1.2.3 Nonlinear download rate amplification

Finally, we describe our results in Section 5 for nonlinear rate amplification with a small error probability.

Slepian-Wolf-style Compression. We begin with any HSS scheme Π for a function class $\mathcal{F}$. Suppose that, sharing a secret $\mathbf{x}$ under Π, each server j has an output share (that is, the output of Eval), z_j. The vector $\mathbf{z}$ of these output shares is a random variable, over the randomness of Share and Eval. Thus, if we repeat this ℓ times with ℓ secrets $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ to get ℓ draws $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_\ell$, we may hope to compress the sequence of $\mathbf{z}_i$'s if, say, $H(\mathbf{z})$ is small.

There are two immediate obstacles to this hope. The first obstacle is that each vector $\mathbf{z}_i$ is split between the k servers, with each server holding only one coordinate. The second obstacle is that the underlying distribution of each $\mathbf{z}_i$ depends on the secret $\mathbf{x}_i$, which is not known to the reconstruction algorithm. Both of these obstacles can be overcome directly by having each server compress its shares individually. This trivially gets around the first obstacle, and it gets around the second because, by t-privacy, the distribution of any one output share does not depend on the secret. However, we can do better.

The first obstacle has a well-known solution, known as *Slepian-Wolf coding.* In Slepian-Wolf coding, a random source $\mathbf{z}$ split between k servers as above can be compressed separately by each server, with download cost for a sequence of length ℓ approaching $\ell \cdot \max_{S \subseteq [k]} H(\mathbf{z}_S | \mathbf{z}_{S^c})$. (Here, $\mathbf{z}_S$

denotes the restriction of $\mathbf{z}$ to the coordinates in S.) Unfortunately, classical Slepian-Wolf coding does not work in the face of the second obstacle, that is if the underlying distribution is unknown.

The most immediate attempt to adapting the classical Slepian-Wolf argument to deal with unknown underlying distributions is to take a large union bound over all $|\mathcal{X}^m|^\ell$ possible sequences of secrets. Unfortunately, this does not work, as the union bound is too big. However, by using the *method of types* (see, e.g., [CT06, Section 11.1]), we are able to reduce the union bound to a manageable size. This results in our main technical theorem of this section, Theorem 12. We instantiate Theorem 12 with 3-server HSS for the AND function, based on 3-party CNF sharing, demonstrating how to beat the impossibility result in Theorem 7 even for a simple and well-motivated instance of HSS.

Symmetric privacy for free. A useful added feature for HSS is having output shares that hide all information other than the output. We refer to this as *symmetric privacy.* The traditional method of achieving this is by "rerandomizing" the output shares. However, this approach conflicts with the compression methodology discussed above. Somewhat unexpectedly, we show (Proposition 4) that our rate-optimized HSS for AND already satisfies the symmetric privacy property.

To give a rough idea why, we start by describing the HSS scheme for AND that we use to instantiate Theorem 12 in an optimal way. In fact, we describe and analyze a generalization to multiplying two inputs a, b in a finite field $\mathbb{F}$ (the AND scheme is obtained by using $\mathbb{F} = \mathbb{F}_2$). The Share function shares each secret using 1-private CNF. Concretely, a is first randomly split into $a = a_1 + a_2 + a_3$ and similarly b, and then server j gets the 4 shares a_i, b_i with $i \neq j$. For defining Eval, we can assign each of the 9 monomials $a_i b_j$ to one of the servers that can evaluate it, and let each server compute the sum of its assigned monomials. It turns out that the monomial assignment for which Theorem 12 yields the best rate is the *greedy* assignment, where each monomial is assigned to the first server who can evaluate it. Using this assignment, the first and last output shares are $y^{(1)} = (a_2 + a_3)(b_2 + b_3) = (a - a_1)(b - b_1)$ and $y^{(3)} = a_1 b_2 + a_2 b_1$. Since $y^{(1)} + y^{(2)} + y^{(3)} = ab$, it suffices to show that the joint distribution of $(y^{(1)}, y^{(3)})$ reveals no information about a, b other than ab.

This can be informally argued as follows. First, viewing $y^{(3)}$ as a randomized function of a_1, b_1 with randomness a_2, b_2, the only information revealed by $y^{(3)}$ about a_1, b_1 is whether $a_1 = b_1 = 0$. Since a_2, b_2 are independent of $(a - a_1)(b - b_1)$, the information about (a, b) revealed by $(y^{(1)}, y^{(3)})$ is equivalent to $(a - a_1)(b - b_1)$ together with the predicate $a_1 = b_1 = 0$. Since $y^{(1)}$ is independent of a, b and is equal to ab conditioned on $a_1 = b_1 = 0$, the latter reveals nothing about a, b other than ab, as required. In the formal proof of symmetric privacy (Proposition 4) we show an explicit bijection between the randomness leading to the same output shares given two pairs of inputs that have the same product.

To complement the above, we observe (Proposition 3) that if we use the natural HSS for multiplication based on Shamir's scheme (namely, locally multiply Shamir shares without rerandomizing), then symmetric privacy no longer holds. Indeed, in this scheme the output shares determine the product of two random degree-1 polynomials with free coefficients a and b respectively. Thus, one can distinguish between the case $a = b = 0$, in which the product polynomial is of the form αX^2, and the case where $a = 0, b = 1$, in which the product polynomial typically has a linear term. Note that the two cases should be indistinguishable, since in both we have $ab = 0$. The insecurity of homomorphic multiplication without share randomization has already been observed in the literature on secure multiparty computation [BGW88].

1.3 Related Work

We already mentioned related work on homomorphic secret sharing, fully homomorphic encryption, private information retrieval, and secure multiparty computation. In the following we briefly survey related work on regenerating codes and communication-efficient secret sharing.

1.3.1 Regenerating codes

Our Shamir-based HSS scheme is inspired by *regenerating codes* [DGW$^+$10], and in particular the work on using Reed-Solomon codes as regenerating codes [SPDC14, GW16, TYB17]. A *Reed-Solomon code* of block length k and degree d is the set $C = \{(p(\alpha_1), \ldots, p(\alpha_k)) : p \in \mathbb{F}[X], \deg(p) \leq d\}$. A *regenerating code*, introduced by [DGW$^+$10] in the context of distributed storage, is a code that allows the recovery of a single erased codeword symbol by downloading not too much information from the remaining symbols. The goal is to minimize the number of bits downloaded from the remaining symbols. Thus, a repair scheme for degree dt Reed-Solomon codes immediately yields an HSS for degree-d polynomials with t-private Shamir-sharing with the same download cost. It turns out that one can indeed obtain download-optimal HSS schemes for low degree polynomials this way from the regenerating codes in [TYB17] (see Corollary 2). However, while this result obtains the optimal download rate of $1 - dt/k$, even for $\ell = 1$, the field size $\mathbb{F}$ must be extremely large: doubly exponential in the number of servers k. Alternatively, if we would like to share secrets over $\mathbb{F}_2$, for example, the upload cost must be huge (see Remark 3), even worse than CNF. Moreover, [TYB17] shows that this is unavoidable if we begin with a regenerating code: any linear repair scheme for Reed-Solomon codes that corresponds to an optimal-rate HSS must have (nearly) such a large field size. In contrast, our results in Section 3 yield Shamir-based HSS with optimal download rate and with reasonable field size and upload cost.

The reason that we are able to do better (circumventing the aforementioned negative result of [TYB17] for Reed-Solomon regenerating codes) is that (a) in HSS we are only required to recover the secret, while in renegerating codes one must be able to recover any erased codeword symbol (corresponding to any given share); (b) we allow the shares to be over a larger field than the secret comes from;[5] and (c) we amortize over $\ell > 1$ instances.

However, even though we cannot use a regenerating code directly, we use ideas from the regenerating codes literature. In particular, our scheme can be viewed as one instantiation of the framework of [GW16] and has ideas similar to those in [TYB17]; again, our situation is simpler particularly due to (a) above.

We mention a few related works that have also used techniques from regenerating codes. First, the work [ACE$^+$21] uses regenerating codes, including a version of the scheme from [GW16], in order to reduce the communication cost per multiplication in secure multiparty computation. Their main result is a logarithmic-factor improvement in the communication complexity for a natural class of MPC tasks compared to previous protocols with the same round complexity.[6] Second, the recent

[5]In the regenerating codes setting, this corresponds to moving away from the MSR (Minimum Storage Regenerating codes) point and towards the MBR (Minimum Bandwidth Regenerating codes) point; see [DRWS11]. To the best of our knowledge, repair schemes for Reed-Solomon codes have not been studied in this setting.

[6]One may ask why [ACE$^+$21] can use a regenerating code while we cannot. The reason is that we are after *optimal* download rate. Indeed, one can obtain nontrivial download rate in our setting using a variant of the scheme in [GW16], which does have a small field size. However, as is necessary for regenerating codes over small fields, the bandwidth of the regenerating code does not meet the so-called *cut-set bound*, and correspondingly the download rate obtained this way is not as good as the optimal $1 - dt/k$ download rate that is achieved with our approach.

work [SW21] studies an extension of regenerating codes (for the special case of Reed-Solomon codes) where the goal is not to compute a single missing symbol but rather any linear function of the symbols. While primarily motivated by distributed storage, that result can be viewed as studying the download cost of HSS for Shamir sharing, in the single-client case where $m = 1$, and restricted to linear functions. One main difference of [SW21] from our work is that in [SW21] the secrets are shared jointly, while in our setting (with several clients) the secrets must be shared independently. Thus [SW21] does not immediately imply any results in our setting. Finally, the work [EGKY21] studies the connection between regenerating codes and proactive secret sharing.

1.3.2 Communication-efficient secret sharing

As noted above, the HSS problem is easier than the general problem of regenerating codes, as we only need to recover the secret(s), rather than any missing codeword symbol (which corresponds to recovering any missing share in the HSS setting). As such, one might hope to get away with smaller field size. In fact, this has been noticed before, and previous work has capitalized on this in the literature on *Communication-Efficient Secret Sharing* (CESS) [HLKB16, HB17, BER17, RKV18]. The simplest goal in this literature is to obtain optimal-download-rate HSS for the special case that $\mathcal{F}$ consists only of the identity function; more complicated goals involve (simultaneously) obtaining the best download rate for any given authorized set of servers (not just $[k]$); and also being able to recover missing shares (as in regenerating codes). Most relevant for us, the simplest goal (and more besides) have been attained, and optimal schemes are known (e.g., [HLKB16]).

However, while related, CESS—even those based on Shamir-like schemes as in [HLKB16]—do not immediately yield results for HSS, or even for HSS-for-concatenation. The main difference is that in CESS, the inputs need not be shared separately. For example, when restricted to the setting of HSS for the identity function, the scheme in [HLKB16] is simply the ℓ-LMSSS described in Remark 1(b), where the ℓ inputs are interpreted as coefficients of the same polynomial and are shared *jointly.*

One exception is the scheme from [HB17], which is directly based on Shamir's scheme (with only one input) over a field $\mathbb{F}$. The scheme is linear, and so it immediately yields an HSS for degree-d polynomials. However, while the download rate approaches optimality as the size of the field $\mathbb{F}$ grows, it is not optimal.[7]

Organization of the paper. In Section 2, we set notation and formally define the notions of HSS and LMSSS that we will use. In Section 3, we present our results for linear HSS. In Section 4, we present our results for black-box rate amplification of additive HSS. Finally, in Section 5, we present our results for rate amplification using nonlinear reconstruction.

2 Preliminaries

Notation. For an integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For an object w in some domain $\mathcal{W}$, we use $|w|$ to denote the number of *bits* required to write down w. That is,

[7]In more detail, the download rate of the t-private, k-server Shamir-based scheme for degree-d polynomials in [HB17] is $\left(\frac{k}{k-dt} + \frac{k^2(k-dt)^2}{4\log_{|\mathbb{B}|}|\mathbb{F}|}\right)^{-1}$, where $\mathbb{B}$ is an appropriate subfield of $\mathbb{F}$. In particular, $\mathbb{F}$ should be exponentially large in k before this rate is near-optimal.

$|w| = \log_2 |\mathcal{W}|$. We will only use this notation when the domain is clear. We generally use bold symbols (like $\mathbf{x}$) to denote vectors.

2.1 Homomorphic Secret Sharing

We consider HSS schemes with m inputs and k servers; we assume that each input is shared independently. We would like to compute functions from a function class $\mathcal{F}$ consisting of functions $f : \mathcal{X}^m \to \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are input and output domains, respectively. Formally, we have the following definition.

Definition 1 (HSS, modified from [BGI+18])**.** *A k-server HSS for a function family $\mathcal{F} = \{f : \mathcal{X}^m \to \mathcal{Y}\}$ is a tuple of algorithms $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ with the following syntax:*

- $\mathsf{Share}(x)$*: On input $x \in \mathcal{X}$ the (randomized) sharing algorithm* Share *outputs k shares $(x^{(1)}, \ldots, x^{(k)})$. We will sometimes write* $\mathsf{Share}(x, r)$ *to explicitly refer to the randomness r used by* Share*. We refer to the $x^{(j)}$ as* input shares*.*
- $\mathsf{Eval}(f, j, (x_1^{(j)}, \ldots, x_m^{(j)}))$*: On input $f \in \mathcal{F}$ (evaluated function), $j \in [k]$ (server index) and $x_1^{(j)}, \ldots, x_m^{(j)}$ (jth share of each input), the evaluation algorithm* Eval *outputs $y^{(j)}$, corresponding to server j's share of $f(x_1, \ldots, x_m)$. We refer to the $y^{(j)}$ as the* output shares*.*
- $\mathsf{Rec}(y^{(1)}, \ldots, y^{(k)})$*: Given $y^{(1)}, \ldots, y^{(k)}$ (list of output shares), the reconstruction algorithm* Rec *computes a final output $y \in \mathcal{Y}$.*

The algorithms $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ should satisfy the following requirements:

- ***Correctness:*** *For any m inputs $x_1, \ldots, x_m \in \mathcal{X}$ and $f \in \mathcal{F}$,*

$$\Pr\left[\mathsf{Rec}\left(y^{(1)}, \ldots, y^{(k)}\right) = f(x_1, \ldots, x_m) : \begin{array}{c} \forall i \in [m] \left(x_i^{(1)}, \ldots, x_i^{(k)}\right) \leftarrow \mathsf{Share}(x_i) \\ \forall j \in [k]\ y^{(j)} \leftarrow \mathsf{Eval}(f, j, (x_1^{(j)}, \ldots, x_m^{(j)})) \end{array}\right] = 1.$$

  *If instead the above probability of correctness is at least α for some $\alpha \in (0,1)$ (rather than being exactly 1), we say that Π is α-*correct*.*

- ***Security:*** *We say that Π is t-*private*, if for every $T \subseteq [k]$ of size $|T| \le t$ and $x, x' \in \mathcal{X}$, the distributions $(x^{(j)})_{j \in T}$ and $((x')^{(j)})_{j \in T}$ are identical, where $\mathbf{x}$ is sampled from* $\mathsf{Share}(x)$ *and $\mathbf{x}'$ from* $\mathsf{Share}(x')$*.*

While the above definition does not refer to computational complexity, in positive results we require by default that all of algorithms are polynomial in their input and output length.

A major theme of this work is amortizing the download cost of HSS over ℓ function evaluations. Informally, there are now ℓ points in $\mathcal{X}^m$, $\mathbf{x}_j = (x_{1,j}, x_{2,j}, \ldots, x_{m,j})$ for each $j \in [\ell]$, and each input $x_{i,j}$ is shared *separately* using Share. The goal is to compute $f_j(\mathbf{x}_j)$ for each $j \in [\ell]$ for some $f_j \in \mathcal{F}$. Formally, we can view this as a special case of Definition 1 applied to the following class $\mathcal{F}^\ell$.

Definition 2 (The class $\mathcal{F}^\ell$)**.** *Given a function class $\mathcal{F}$ that maps $\mathcal{X}^m$ to $\mathcal{Y}$, we define $\mathcal{F}^\ell$ to be the function class that maps $\mathcal{X}^{\ell m}$ to $\mathcal{Y}^\ell$ given by*

$$\mathcal{F}^\ell := \{(x_{i,j})_{i \in [m], j \in [\ell]} \mapsto (f_1(\mathbf{x}_1), \ldots, f_\ell(\mathbf{x}_\ell)) \ : \ f_1, \ldots, f_\ell \in \mathcal{F}\}.$$

Computational HSS. In this work we will be primarily interested in information-theoretic HSS as in Definition 1. However, in Section 4 we will also be interested in *computationally secure* HSS schemes, where the security requirement is relaxed to hold against computationally bounded distinguishers. A formal definition appears in Appendix A.

We will be particularly interested in HSS schemes whose sharing and/or reconstruction functions are linear functions over a finite field, defined as follows.

Definition 3 (Linear HSS). *Let $\mathbb{F}$ be a finite field. We say that an HSS scheme $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ has* linear reconstruction *over $\mathbb{F}$ if $\mathcal{Y} = \mathbb{F}^b$ for some integer $b \geq 1$; $\mathsf{Eval}(f, j, \mathbf{x}^{(j)})$ outputs $y^{(j)} \in \mathbb{F}^{b_j}$ for integer $b_j \geq 0$; and $\mathsf{Rec} : \mathbb{F}^{\sum_j b_j} \to \mathbb{F}^b$ is an $\mathbb{F}$-linear map. We say that Π has* additive reconstruction *over $\mathbb{F}$, or simply that Π is* additive*, if $b = b_j = 1$ for all j and $\mathsf{Rec}(y^{(1)}, \ldots, y^{(k)}) = y^{(1)} + \ldots + y^{(k)}$.*

Finally, we say that Π is linear *if it has linear reconstruction and in addition, $\mathcal{X} = \mathbb{F}$ and $\mathsf{Share}(x, \mathbf{r})$ is an $\mathbb{F}$-linear function of x and a random vector $\mathbf{r}$ with i.i.d. uniform entries in $\mathbb{F}$. Notice that we never require Eval to be linear.*

The main focus of this work is on the communication complexity of an HSS scheme. We formalize this with the following definitions.

Definition 4 (Upload and download costs and rate). *Let k, t be integers and let $\mathcal{F} = \{f : \mathcal{X}^m \to \mathcal{Y}\}$ be a function class. Let Π be a k-server t-private HSS for $\mathcal{F}$. Suppose that the input shares for Π are $x_i^{(j)}$ for $i \in [m], j \in [k]$, and that the output shares are $y^{(1)}, \ldots, y^{(k)}$. We define*

- *The* upload cost *of Π,* $\mathrm{UploadCost}(\Pi) = \sum_{i=1}^{m} \sum_{j=1}^{k} |x_i^{(j)}|$.
- *The* download cost *of Π,* $\mathrm{DownloadCost}(\Pi) = \sum_{j=1}^{k} |y^{(j)}|$.
- *The* download rate *(or just* rate*) of Π,*

$$\mathrm{Rate}(\Pi) = \frac{\log_2 |\mathcal{Y}|}{\mathrm{DownloadCost}(\Pi)}$$

Symmetrically private HSS. Several applications of HSS motivate a *symmetrically private* variant in which the output shares $(y^{(1)}, \ldots, y^{(k)})$ reveal no additional information about the inputs beyond the output of f. Any HSS with *linear reconstruction* (Definition 3) can be modified to meet this stronger requirement without hurting the download rate (and with only a small increase to the upload cost) via a simple randomization of the output shares. We further discuss this variant in Section 5.

Private information retrieval. Some of our HSS results have applications to *private information retrieval* (PIR) [CGKS98]. A t-private k-server PIR protocol allows a client to retrieve a single symbol from a database in $\mathcal{Y}^N$, which is replicated among the servers, such that no t servers learn the identity of the retrieved symbol. Note that such a PIR protocol reduces to a t-private k-server HSS scheme for the family $\mathsf{ALL}_{\mathcal{Y}}$ of *all* functions $f : [N] \to \mathcal{Y}$, where the number of inputs is $m = 1$. Indeed, in order to retrieve the i'th symbol $f(i)$ from a database represented by the function f, the client may use an HSS to share the input $x = i$ among the k servers; each server computes Eval on their input share and the database and sends the output share back to the client; the client then runs Rec in order to obtain $f(i)$. The download rate and upload cost of a PIR protocol are defined as in Definition 4.

2.2 Linear Secret Sharing Schemes

In this section we define and give common examples of (information theoretic) linear secret sharing schemes (LSSS), with secrets from some finite field $\mathbb{F}$. We consider a generalized *linear multi-secret sharing scheme* (LMSSS) notion, which allows one to share multiple secrets.

Definition 5 (LMSSS)**.** *Let $\Gamma, \mathcal{T} \subseteq 2^{[k]}$ be monotone (increasing and decreasing, respectively)*[8] *collections of subsets of $[k]$, so that $\mathcal{T} \cap \Gamma = \emptyset$. A k-party ℓ-LMSSS $\mathcal{L}$ over a field $\mathbb{F}$ with access structure Γ and adversary structure $\mathcal{T}$ is specified by numbers $e, b_1, \ldots, b_k$ and a linear mapping $\mathsf{Share} : \mathbb{F}^{\ell} \times \mathbb{F}^{e} \to \mathbb{F}^{b_1} \times \ldots \times \mathbb{F}^{b_k}$ so that the following holds.*

- *Correctness: For any qualified set $Q = \{j_1, \ldots, j_m\} \in \Gamma$ there exists a linear reconstruction function $\mathsf{Rec}_Q : \mathbb{F}^{b_{j_1}} \times \ldots \times \mathbb{F}^{b_{j_m}} \to \mathbb{F}^{\ell}$ such that for every $\mathbf{x} \in \mathbb{F}^{\ell}$ we have that $\Pr_{\mathbf{r} \in \mathbb{F}^e}[\mathsf{Rec}_Q(\mathsf{Share}(\mathbf{x}, \mathbf{r})_Q) = \mathbf{x}] = 1$, where $\mathsf{Share}(\mathbf{x}, \mathbf{r})_Q$ denotes the restriction of $\mathsf{Share}(\mathbf{x}, \mathbf{r})$ to its entries indexed by Q.*
- *Privacy: For any unqualified set $U \in \mathcal{T}$ and secrets $\mathbf{x}, \mathbf{x}' \in \mathbb{F}^{\ell}$, the random variables $\mathsf{Share}(\mathbf{x}, \mathbf{r})_U$ and $\mathsf{Share}(\mathbf{x}', \mathbf{r})_U$, for uniformly random $\mathbf{r} \in \mathbb{F}^e$, are identically distributed.*

*If $\mathcal{T}$ contains all sets of size at most t (and possibly more), we say that $\mathcal{L}$ is t-*private*. If $\ell = 1$ we simply call $\mathcal{L}$ an* LSSS*, and we refer to the ℓ-LMSSS obtained via ℓ independent repetitions of $\mathcal{L}$ as* ℓ instances of $\mathcal{L}$*. Finally, we define the* information rate *of $\mathcal{L}$ to be $\ell/(b_1 + \ldots + b_k)$.*

Additive sharing is an important example of an LSSS.

Example 1 (Additive sharing)**.** *The* additive sharing *of a secret $x \in \mathbb{F}$ is a $(k-1)$-private LSSS with $\Gamma = [k]$, $e = k - 1$ and $b_j = 1$ for all $j \in [k]$. It is defined as follows.*

- ***Sharing.*** *Let $\mathsf{Share}(x, \mathbf{r}) = (r_1, r_2, \ldots, r_{k-1}, x - r_1 - \ldots - r_{k-1})$. Note that the shares are uniformly distributed over $\mathbb{F}^k$ subject to the restriction that they add up to x.*
- ***Reconstruction.*** *Let $\mathsf{Rec}_{[k]}(x^{(1)}, \ldots, x^{(k)}) = x^{(1)} + \ldots + x^{(k)}$.*

We now define two standard LSSS's and associated ℓ-LMSSS's we will use in this work: the so-called "CNF scheme" [ISN89] (also referred to as replicated secret sharing) and Shamir's scheme [Sha79].

Definition 6 (t-private CNF sharing)**.** *The t-private k-party* CNF sharing *of a secret $x \in \mathbb{F}$ is an LSSS with parameters $e = \binom{k}{t} - 1$ and $b_j = \binom{k-1}{t}$ for all $j \in [k]$. (We use t-*CNF *when k is clear from the context.) It is defined as follows.*

- ***Sharing.*** *Using a random vector $\mathbf{r} \in \mathbb{F}^e$, we first* additively *share x by choosing $\binom{k}{t}$ random elements of $\mathbb{F}$, x_T, so that $x = \sum_{T \subseteq [k] : |T| = t} x_T$. Then we define $\mathsf{Share}(x, \mathbf{r})_j = (x_T)_{j \notin T}$ for $j \in [k]$.*
- ***Reconstruction.*** *Any $t+1$ parties together hold all of the additive shares x_T, and hence can recover x. This defines Rec_Q for $|Q| > t$.*

We note that there is a trivial ℓ-LMSSS variant of t-private CNF sharing, as per Definition 5, which shares ℓ secrets with ℓ independent instances of CNF sharing.

[8] We say that Γ and $\mathcal{T}$ are monotone (increasing and decreasing, respectively) if $Q \subseteq Q'$ and $Q \in \Gamma$ then $Q' \in \Gamma$; and if $T' \subseteq T$ and $T \in \mathcal{T}$ then $T' \in \mathcal{T}$.

Definition 7 (t-private Shamir sharing)**.** *Let $\mathbb{F}$ be a finite field and let $\mathbb{E} \supseteq \mathbb{F}$ be an extension field (typically, the smallest extension field so that $|\mathbb{E}| > k$), and suppose that $s = [\mathbb{E} : \mathbb{F}]$ is the degree of $\mathbb{E}$ over $\mathbb{F}$. Fix distinct evaluation points $\alpha_0, \alpha_1, \ldots, \alpha_k \in \mathbb{E}$. The t-private, k-party* Shamir sharing *of a secret $x \in \mathbb{F}$ (with respect to $\mathbb{E}$ and the α_i's) is an LSSS with parameters $e = t \cdot s$ and $b_i = s$ for all $i \in [k]$, defined as follows.*

- ***Sharing****. Let $x \in \mathbb{F}$ and let $\mathbf{r} \in \mathbb{F}^{ts}$. We may view $\mathbf{r}$ as specifying t random elements of $\mathbb{E}$, and we use $\mathbf{r}$ to choose a random polynomial $p \in \mathbb{E}[X]$ so that $\deg(p) \leq t$ and so that $p(\alpha_0) = x$. Then $\mathsf{Share}(x, \mathbf{r})_j = p(\alpha_j)$.*
- ***Reconstruction****. Any $t + 1$ parties together can obtain $t + 1$ evaluation points of the random polynomial p, and hence can recover $x = p(\alpha_0)$ by polynomial interpolation. This constitutes the* Rec *function.*

Remark 1 (ℓ-LMSSS variants of Shamir sharing)**.** *The definition above is for an LSSS (1-LMSSS). There are several ℓ-LMSSS varints of t-private Shamir sharing. In particular:*

(a) The first variant is the trivial ℓ-LMSSS variant of t-private Shamir sharing where each of ℓ secrets are shared independently. As per Definition 5, we refer to this as "ℓ instances of t-private Shamir sharing".

(b) The second (and third) variants are where $\ell = k - t$ secrets are encoded as different evaluation points of a polynomial with degree $\ell + t - 1$ (requiring $|\mathbb{E}| > 2k - \ell$), or, alternatively, different coefficients (requiring $|\mathbb{E}| > k$). These two ℓ-LMSSS variants of Shamir sharing (the first of which is sometimes referred to as the Franklin-Yung scheme [FY92]) have an information rate of $\frac{\ell \log |\mathbb{F}|}{k \log |\mathbb{E}|} = \frac{1 - t/k}{s}$.

Local share conversion. Informally, local share conversion allows the parties to convert from one LMSSS to another without communication. That is, the conversion maps *any* valid sharing of $\mathbf{x}$ using a source scheme $\mathcal{L}$ to *some* (not necessarily random) valid sharing of $\mathbf{x}$ (more generally, some function $\psi(\mathbf{x})$), according to the target scheme $\mathcal{L}'$. Formally, we have the following definition, which extends the definitions of [CDI05, BIKO12] to multi-secret sharing.

Definition 8 (Local share conversion)**.** *Suppose that $\mathcal{L} = (\mathsf{Share}, \mathsf{Rec})$ is a k-party ℓ-LMSSS with parameters $(e, b_1, \ldots, b_k)$, and suppose that $\mathcal{L}' = (\mathsf{Share}', \mathsf{Rec}')$ is a k-party ℓ'-LMSSS with parameters $(e', b'_1, \ldots, b'_k)$. Let $\psi : \mathbb{F}^\ell \to \mathbb{F}^{\ell'}$. A* local share conversion *from $\mathcal{L}$ to $\mathcal{L}'$ with respect to ψ is given by functions $\varphi_i : \mathbb{F}^{b_i} \to \mathbb{F}^{b'_i}$ for $i \in [k]$, so that for any secret $\mathbf{x} \in \mathbb{F}^\ell$, for any $\mathbf{r} \in \mathbb{F}^e$, there is some $\mathbf{r}' \in \mathbb{F}^{e'}$ so that*

$$(\varphi_1(\mathsf{Share}(\mathbf{x}, \mathbf{r})_1), \ldots, \varphi_k(\mathsf{Share}(\mathbf{x}, \mathbf{r})_k)) = \mathsf{Share}'(\psi(\mathbf{x}), \mathbf{r}').$$

If there is a local share conversion from $\mathcal{L}$ to $\mathcal{L}'$ with respect to ψ, we say that $\mathcal{L}$ is locally convertible with respect to ψ *to $\mathcal{L}'$. When ψ is the identity map, we just say that $\mathcal{L}$ is locally convertible to $\mathcal{L}'$.*

It was shown in [CDI05] that t-private CNF sharing can be locally converted to any LSSS $\mathcal{L}'$ which is (at least) t-private. Formally:

Theorem 1 ([CDI05])**.** *Let $\mathcal{L}$ be the t-private k-party CNF LSSS over a finite field $\mathbb{F}$ (Definition 6). Then $\mathcal{L}$ is locally convertible (with respect to the identity map ψ) to any t-private LSSS $\mathcal{L}'$ over $\mathbb{F}$.*

We will use a natural extension of this idea: that ℓ instances of k-server CNF can be jointly locally converted to any k-server ℓ-LMSSS with appropriate adversary structure.

Corollary 1. *Let $\mathcal{L}$ be the k-party ℓ-LMSSS given by ℓ instances of t-CNF secret sharing over $\mathbb{F}$ (Definition 6). Then $\mathcal{L}$ is locally convertible to any t-private k-party ℓ-LMSSS $\mathcal{L}'$ over $\mathbb{F}$.*

Proof. Observe that we may obtain an LSSS $\mathcal{L}'_i$ for each $i \in [\ell]$ from $\mathcal{L}'$ by considering the LSSS that uses $\mathcal{L}'$ to share $(0, \ldots, 0, x_i, 0, \ldots, 0)$, where x_i is in the $i'th$ position. Note that each $\mathcal{L}'_i$ is also t-private, by definition of an LMSSS. Now, consider the secret-sharing scheme $\mathcal{L}_i$ that shares $(0, 0, \ldots, 0, x_i, 0, \ldots, 0)$ using $\mathcal{L}$; this is just the standard t-CNF LSSS. Thus, we may apply Theorem 1 to locally convert $\mathcal{L}_i$ to $\mathcal{L}'_i$ for each $i \in [\ell]$. Finally, each party adds up element-wise its shares of all schemes $\mathcal{L}'_i$ to obtain, by linearity, a sharing of $(x_1, \ldots, x_\ell)$ according to $\mathcal{L}'$. $\square$

3 Linear HSS for Low-Degree Polynomials

In this section, we give both positive and negative results for linear HSS for low-degree multivariate polynomials. Before we get into our results, we state a few useful definitions.

Definition 9. *Let $m > 0$ be an integer and let $\mathbb{F}$ be a finite field. We define*

$$\mathsf{POLY}_{d,m}(\mathbb{F}) = \{f \in \mathbb{F}[X_1, \ldots, X_m] : \deg(f) \leq d\}$$

to be the class of all m-variate degree-at-most-d polynomials over $\mathbb{F}$. When m and $\mathbb{F}$ are clear from context, we will just write POLY_d to refer to $\mathsf{POLY}_{d,m}(\mathbb{F})$.

The class POLY_d^ℓ may be interesting even when $d = 1$. In this case, the problem can be reduced to "HSS for concatenation." That is, we are given ℓ secrets $x_1, \ldots, x_\ell \in \mathbb{F}$, shared *separately*, and we must locally convert these shares to small joint shares of $\mathbf{x} = (x_1, \ldots, x_\ell)$. (To apply this towards HSS for POLY_1^ℓ, first locally compute shares of the ℓ outputs from shares of the inputs, and then apply HSS for concatenation to reconstruct the outputs.) Formally, we have the following definition.

Definition 10 (HSS for concatenation)**.** *Let $\mathcal{X}$ be any alphabet and let $\mathcal{Y} = \mathcal{X}^\ell$. We define $f : \mathcal{X}^\ell \to \mathcal{Y}$ to be the identity map, and $\mathsf{CONCAT}_\ell(\mathcal{X}) = \{f\}$. We refer to an HSS for $\mathsf{CONCAT}_\ell(\mathcal{X})$ as* HSS for concatenation*.*

Note that we view $m = \ell$ as the number of inputs, and so an HSS for concatenation must share each input $x_i \in \mathcal{X}$ independently. Also note that a linear HSS for $\mathsf{CONCAT}_\ell(\mathbb{F})$ is equivalent to a linear HSS for $\mathsf{POLY}_{1,1}(\mathbb{F})^\ell$. While HSS for concatenation seems like a basic primitive, to the best of our knowledge it has not been studied before. In this work we will need HSS-for-concatenation with specific kinds of Share functions. Without this restriction, the problem has a trivial optimal solution discussed in the following remark.

Remark 2 (Trivial constructions of HSS-for-concatenation)**.** *Given any ℓ-LMSSS $\mathcal{L}$, there is a simple construction of an HSS-for-concatenation $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ with the same information rate as $\mathcal{L}$. In more detail, suppose that $\mathcal{L}$ has share function Share' and reconstruction function Rec'. Given input $x \in \mathcal{X}$, we define Share by sharing $(x, 0, \ldots, 0), \ldots, (0, \ldots, 0, x)$ separately using Share'. Now, suppose the servers are given shares $\mathbf{y}_i^{(j)}$ of inputs x_i for $i \in [\ell]$ and $j \in [k]$. Then we define*

Eval *so that server j's output share is $z^{(j)} = \sum_i y_i^{(j)}$, where $y_i^{(j)}$ is the part of $\mathbf{y}_i^{(j)}$ corresponding to the i'th instance of* Share'. *Finally, we define*

$$\mathsf{Rec}(z^{(1)}, \ldots, z^{(k)}) = \mathsf{Rec}'_{[k]}(z^{(1)}, \ldots, z^{(k)}).$$

The correctness of this scheme follows from the linearity of $\mathsf{Rec}'_{[k]}$.

While this simple construction results in optimal linear HSS-for-concatenation, our applications depend on HSS-for-concatenation with particular Share *functions (like the LSSS versions of t-CNF or Shamir). Unfortunately, the share functions arising from natural ℓ-LMSSS schemes (like the ℓ-LMSSS version of Shamir in Remark 1(b)) will not work in our applications (c.f. Remark 4).*

3.1 Constructions of Linear HSS for Low-Degree Polynomials

In this section, we give two constructions of HSS for POLY_d^ℓ, one based on CNF sharing in Section 3.1.1 and one on Shamir sharing in Section 3.1.2; we will show later in Section 3.2 that both of these have optimal download rate among linear schemes for sufficiently large ℓ. We further compare the two schemes in Section 3.1.3, and we work out an application to PIR in Section 3.1.4.

3.1.1 Linear HSS from CNF sharing

Our main result in this section is a tight connection between HSS-for-concatenation (that is, for CONCAT_ℓ) and linear codes. More precisely, we show in Theorem 2 that the best trade-off between t and the download rate R for HSS-for-concatenation (assuming that the output of Eval has the same size for each party) is captured by the best trade-off between distance and rate for certain linear codes. Then we extend this result to obtain constructions of HSS for POLY_d^ℓ.

The basic idea is to begin with CNF sharing, and then use Corollary 1 to convert the CNF shares to shares of some ℓ-LMSSS $\mathcal{L}$. The final outputs of Eval will be the shares of $\mathcal{L}$, so the total download rate of the resulting HSS scheme will be the information rate of $\mathcal{L}$. Thus, we need to study the best t-private ℓ-LMSSS with access structure $[k]$. To do this, we utilize a connection between such LMSSS and linear error correcting codes.

Let $C \subseteq (\mathbb{F}^b)^k$ be an $\mathbb{F}$-linear subspace of $(\mathbb{F}^b)^k$. We call C an *$\mathbb{F}$-linear code with alphabet $\mathbb{F}^b$ and block-length k.*[9] We define the *rate* of such a code C to be

$$\mathrm{Rate}(C) = \frac{\dim_{\mathbb{F}}(C)}{bk}$$

and the distance to be

$$\mathrm{Dist}(C) = \min_{c \neq c' \in C} |\{i \in [k] \,:\, c_i \neq c'_i\}|.$$

It is well-known that LMSSSs can be obtained from error correcting codes with good dual distance (see, e.g., [Mas95, CCG+07]). However, in order to construct HSS we are interested only in LMSSS with access structure $\Gamma = [k]$, which results in a particularly simple correspondence. We record this correspondence in the lemma below. (The special case where $b = 1$ follows from [GM10].)

[9]Codes that are linear over a field $\mathbb{F}$ and have alphabet $\mathbb{F}^b$ show up often in coding theory; for example, folded codes and multiplicity codes both have this feature. Moreover, any linear code over $\mathbb{F}_{q^b}$ can be viewed as an $\mathbb{F}_q$-linear code over $\mathbb{F}_q^b$, by replacing each symbol $\alpha \in \mathbb{F}_{q^b}$ with an appropriate vector in $\mathbb{F}_q^b$.

Lemma 1 (Generalizing Theorem 1 of [GM10]). *Let $\ell < bk$. There is a t-private k-party ℓ-LMSSS $\mathcal{L}$ over $\mathbb{F}$ with shares in $\mathbb{F}^b$ and access structure $\Gamma = [k]$ (in particular, with rate $\frac{\ell}{kb}$) if and only if there is an $\mathbb{F}$-linear code $C \subseteq (\mathbb{F}^b)^k$ of information rate $R \geq \frac{\ell}{kb}$ and distance at least $t+1$.*

Proof. We begin with some notation. Given a vector $\mathbf{v} \in \mathbb{F}^{bk}$, we will write $\mathrm{chop}(\mathbf{v}) = (\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(k)})$, where each $\mathbf{v}^{(i)} \in \mathbb{F}^b$, to denote the natural way of viewing $\mathbf{v}$ as an element of $(\mathbb{F}^b)^k$. Similarly, given $H \in \mathbb{F}^{e \times bk}$, we write $H = [H^{(1)} | H^{(2)} | \cdots | H^{(k)}]$ where $H^{(i)} \in \mathbb{F}^{e \times b}$.

Given a matrix $H \in \mathbb{F}^{e \times kb}$, we may define a corresponding $\mathbb{F}$-linear code $C_H \subset (\mathbb{F}^b)^k$ by

$$C_H := \{\mathrm{chop}(\mathbf{c}) \,:\, \mathbf{c} \in \mathbb{F}^{bk} \,:\, H\mathbf{c} = 0\}. \tag{1}$$

Conversely, any $\mathbb{F}$-linear code $C \subseteq (\mathbb{F}^b)^k$ can be written as C_H for some (not necessarily unique) $H \in \mathbb{F}^{e \times bk}$ for some e.

Now we prove the "only if" direction of the lemma. Let $C \subseteq (\mathbb{F}^b)^k$ be a code as in the lemma statement. By assumption, the dimension of the code is $Rkb \geq \ell$. Since C is linear, we may write $C = C_H$ as in (1) for some full-rank matrix $H \in \mathbb{F}^{e \times bk}$, where $e = bk(1-R)$. Let $G \in \mathbb{F}^{bk \times \ell}$ be any matrix so that $[G | H^T] \in \mathbb{F}^{bk \times (\ell + e)}$ has rank $\ell + e$. Notice that this exists because H is full-rank and

$$e + \ell = bk(1-R) + \ell \leq bk(1-R) + bkR = bk.$$

Then consider the ℓ-LMSSS $\mathcal{L}$ given by

$$\mathsf{Share}(\mathbf{x}, \mathbf{r}) = \mathrm{chop}(G\mathbf{x} + H^T\mathbf{r}),$$

where $\mathbf{x} \in \mathbb{F}^\ell$ contains the secrets and $\mathbf{r} \in \mathbb{F}^e$ is a uniformly random vector. We claim that $\mathcal{L}$ is a t-private ℓ-LMSSS with access structure $\Gamma = [k]$. To see that it is t-private, note that the distance of C_H implies that there is no vector $\mathbf{v} \in \mathbb{F}^{bk}$ so that $\mathrm{chop}(\mathbf{v})$ has weight at most t (that is, at most t of the k "chunks" of $\mathbf{v}$ are nonzero) and so that $H\mathbf{v} = 0$. This implies that for any $i_1, \ldots, i_t \in [k]$, the matrix $[H^{(i_1)} | H^{(i_2)} | \ldots | H^{(i_t)}]$ is full rank. This in turn implies that any t elements of $\mathrm{chop}(H^T\mathbf{r})$ are uniformly random, which implies that any t elements of $\mathsf{Share}(\mathbf{x}, \mathbf{r})$ are uniformly random. Thus $\mathcal{L}$ is t-private. The fact that $\mathcal{L}$ admits access structure $\Gamma = [k]$ follows from the fact that $[G | H^T]$ is full rank.

The "if" direction follows similarly. Suppose we have an ℓ-LMSSS $\mathcal{L}$ as in the theorem statement. From the definition of LMSSS, we may write $\mathsf{Share}(\mathbf{x}, \mathbf{r}) = \mathrm{chop}(G\mathbf{x} + H^T\mathbf{r})$ for some matrix $G \in \mathbb{F}^{\ell \times bk}$ and some matrix $H \in \mathbb{F}^{bk \times e}$ for some e. Without loss of generality, we may assume that H has full rank, since otherwise we may replace it with a full rank matrix with fewer rows. Since $\mathcal{L}$ has access structure $\Gamma = [k]$, the matrix $[G | H^T]$ must have rank $\ell + e$. This implies that $e \leq bk - \ell$. Consider the code C_H defined by H. The rate of C_H is $(bk - e)/(bk) \geq \ell / bk$, which is the information rate of $\mathcal{L}$. Finally, by the same logic as above, the fact that $\mathcal{L}$ is t-private implies that C_H has distance at least t. $\square$

With the connection to error correcting codes established, we can now use well-known constructions of error correcting codes in order to obtain good HSS schemes. We begin with a statement for HSS for concatenation (Definition 10). Then we will use share conversion from t-private CNF to dt-private CNF (Lemma 2 below) to extend this to POLY_d^ℓ for $d > 1$.

Theorem 2 (Download rate of HSS-for-concatenation is captured by the distance of linear codes). *Let ℓ, k be integers, and let $\mathbb{F}$ be a finite field. Suppose that $b > \ell / k$.*

There is a t-private, k-server linear HSS for $\mathsf{CONCAT}_\ell(\mathbb{F})$ with output shares in $\mathbb{F}^b$ (and hence with download rate $\ell/(kb)$) if and only if there is an $\mathbb{F}$-linear code $C \in (\mathbb{F}^b)^k$ with rate at least $\ell/(kb)$ and distance at least $t+1$.

Further, for the "if" direction, the HSS scheme guaranteed by the existence of C uses t-CNF sharing and has upload cost $k\ell\binom{k-1}{t}\log_2|\mathbb{F}|$.

Proof. For the "if" direction (a code C implies an HSS-for-concatenation Π), we follow the outline sketched above. Our goal is to share each input x_i independently with CNF sharing and to reconstruct $\mathbf{x} = (x_1, \dots, x_\ell)$ from the outputs of Eval. Let C be a code as in the theorem statement, and define an HSS $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ as follows.

- **Sharing.** Let $\mathsf{Share}(x) = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(k)})$ be t-CNF sharing of x, as in Definition 6.
- **Evaluation.** Given t-CNF shares $\mathbf{x}^{(j)} = (\mathbf{x}_1^{(j)}, \dots, \mathbf{x}_\ell^{(j)})$ for each $j \in [k]$, we use Corollary 1 to locally convert these to shares $\mathbf{y}^{(j)} \in \mathbb{F}^b$ of an ℓ-LMSSS $\mathcal{L}$ for the secret $\mathbf{x} = (x_1, \dots, x_\ell)$, where $\mathcal{L}$ is the ℓ-LMSSS that is guaranteed by Lemma 1 and the existence of the code C. We define $\mathsf{Eval}(f, j, \mathbf{x}^{(j)}) = \mathbf{y}^{(j)}$.
- **Reconstruction.** By construction, we have
$$\mathsf{Rec}_{[k]}(\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(k)}) = \mathbf{x},$$
where $\mathsf{Rec}_{[k]}$ is the reconstruction algorithm from $\mathcal{L}$. This defines the reconstruction algorithm Rec for Π.

The parameters for the "if" and "further" parts of the theorem follow by tracing the parameters through Corollary 1 and Lemma 1.

Now we prove the "only if" direction. Suppose that $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ is an HSS as in the theorem statement. Consider the ℓ-LMSSS $\mathcal{L}$ that shares $\mathbf{x} = (x_1, \dots, x_\ell) \in \mathbb{F}^\ell$ as $\mathsf{Share}'(\mathbf{x}, \mathbf{r}) = (\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(k)})$, where $\mathbf{y}^{(j)}$ is the j'th output of Eval when $\mathbf{x}$ is shared using Share. Since $\mathsf{Rec}(\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(k)}) = \mathbf{x}$ is linear, this secret sharing scheme has a linear reconstruction algorithm. By [Bei96, Claims 4.3, 4.7, 4.9], any secret sharing scheme with a linear reconstruction algorithm is equivalent to a scheme with a linear share function; this argument can be extended to ℓ-LMSSS. Thus there is a linear share function Share'' so that $\mathcal{L} = (\mathsf{Share}'', \mathsf{Rec})$ forms a t-private, k-party ℓ-LMSSS with information rate $\ell/(bk)$. By Lemma 1, there is a code $C \subseteq (\mathbb{F}^b)^k$ with distance at least $t+1$ and rate at least $\ell/(bk)$. This establishes the "only if" direction. □

We can easily extend the "if" direction (the construction of HSS) to $\mathsf{POLY}^\ell_{d,m}(\mathbb{F})$ by using the following lemma, which shows that t-CNF shares of d secrets can be locally converted to valid dt-CNF shares of the product of the secrets via a suitable assignment of monomials to servers. A similar monomial assignment technique was used in the contexts of communication complexity [BGKL03], private information retrieval [BIK05], and secure multiparty computation [Mau06].

Lemma 2. *Let $\mathcal{L}$ be the d-LMSSS that is given by d instances of t-private, k-party CNF sharing. Let $\mathcal{L}'$ be the LSSS given by dt-private, k-party CNF sharing. Let $\psi(x_1, \dots, x_d) = \prod_{i=1}^d x_i$. Then $\mathcal{L}$ is locally convertible to $\mathcal{L}'$ with respect to ψ.*

Proof. We describe the maps $\varphi_i : \mathbb{F}^{d\binom{k-1}{t}} \to \mathbb{F}^{\binom{k-1}{dt}}$ for each $i \in [d]$ that will define the local share conversion. Suppose that the original secrets to be shared with $\mathcal{L}$ are $x_1, \dots, x_d \in \mathbb{F}$. Under $\mathcal{L}$, party j's share consists of $\mathbf{y}^{(j)} = (x_{i,S} : i \in [d], j \notin S)$, where for $S \subseteq [k]$ of size t, the random variables $x_{i,S} \in \mathbb{F}$ are uniformly random so that $x_i = \sum_S x_{i,S}$ for all $i \in [d]$.

For each set $W \subseteq [k]$ of size $|W| \leq dt$, define

$$y_W = \sum_{S_1,\ldots,S_d : \bigcup_i S_i = W} \prod_{i=1}^{d} x_{i,S_i},$$

where the sum is over all sets $S_1, \ldots, S_d \subseteq [k]$ of size t whose union is exactly W. Notice that

$$\begin{aligned} \sum_{W \subseteq [k], |W| \leq dt} y_W &= \sum_{S_1,\ldots,S_d \subset [k] : |S_i| = t} \prod_{i=1}^{d} x_{i,S_i} \\ &= \prod_{i=1}^{d} \left(\sum_{S \subset [k] : |S| = t} x_{i,S} \right) = \prod_{i=1}^{d} x_i = \psi(\mathbf{x}). \end{aligned}$$

Now, for all $W \subset [k]$ with $|W| \leq dt$, arbitrarily assign to W some set $T = T(W) \subset [k]$ of size exactly dt, so that $W \subseteq T$. For $T \subseteq [k]$ of size exactly dt, define

$$z_T = \sum_{W : T = T(W)} y_W,$$

where the sum is over all sets $W \subseteq [k]$ of size at most dt so that $T(W)$ is equal to T. By the above, we have

$$\sum_{T \subseteq [k] : |T| = dt} z_T = \psi(\mathbf{x}).$$

Further, each party j can locally compute z_T for each T so that $j \notin T$. This is because z_T requires knowledge only of $x_{i,S}$ for $i \in [d]$ and for $S \subseteq T$. If $j \notin T$, then $j \notin S$, and therefore $x_{i,S}$ appears as part of the share $\mathbf{y}^{(j)}$. Thus, we define

$$\varphi_j(\mathbf{y}^{(j)}) = (z_T : T \subseteq [k], |T| = dt, j \notin T),$$

and the shares $\varphi_j(\mathbf{y}^{(j)})$ are legitimate shares of $\psi(\mathbf{x})$ under $\mathcal{L}'$. □

Using Lemma 2 along with Theorem 2 implies the following extension to POLY_d^ℓ.

Theorem 3 (Download rate of HSS-for-polynomials follows from distance of linear codes)**.** *Let ℓ, t, k, d, m be integers, and let $\mathbb{F}$ be a finite field. Suppose that for some integer $b > \ell/k$, there is an $\mathbb{F}$-linear code $C \in (\mathbb{F}^b)^k$ with rate at least $\ell/(kb)$ and distance at least $dt + 1$. Then there is a t-private, k-server linear HSS for $\mathsf{POLY}_{d,m}(\mathbb{F})^\ell$ with download rate $\ell/(kb)$ and upload cost $k\ell m\binom{k-1}{t} \log_2 |\mathbb{F}|$.*

Proof. By linearity, we may assume without loss of generality that the function $f \in \mathsf{POLY}_{d,m}(\mathbb{F})^\ell$ is a vector of ℓ *monomials* given by $f_r : \mathbb{F}^m \to \mathbb{F}$ for $r \in [\ell]$, so that that f_r is a monomial of degree at most d. We define a t-private, k-server linear HSS $\Pi' = (\mathsf{Share}', \mathsf{Eval}', \mathsf{Rec}')$ for $\mathsf{POLY}_{d,m}(\mathbb{F})^\ell$ with download rate $\ell/(kb)$ as follows.

Let $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ be the dt-private, k-server linear HSS for $\mathsf{CONCAT}_\ell(\mathbb{F})$ with output shares in $\mathbb{F}^b$ and download rate $\ell/(kb)$ that is guaranteed by Theorem 2. Consider $f \in \mathsf{POLY}_{d,m}(\mathbb{F})^\ell$ of the form described above. For $r \in [\ell]$, let $\varphi_{1,r}, \ldots, \varphi_{k,r}$ be the local share conversion functions guaranteed by Lemma 2 for the monomial f_r.

- **Sharing.** The Share' function is t-CNF sharing. Suppose that we share the inputs $x_{i,r}$ for $i \in [m]$ and $r \in [\ell]$ as shares $\mathbf{y}_r^{(j)} \in \mathbb{F}^{m\binom{k-1}{t}}$ for each party $j \in [k]$ and each $r \in [\ell]$. Let $\mathbf{y}^{(j)} = (\mathbf{y}_1^{(j)}, \dots, \mathbf{y}_\ell^{(j)})$.
- **Evaluation.** Define $\mathbf{z}_j$ and Eval' by:
$$\mathbf{z}^{(j)} = \mathsf{Eval}'(f, j, \mathbf{y}^{(j)}) = \mathsf{Eval}(g, j, (\varphi_{j,1}(\mathbf{y}_1^{(j)}), \dots, \varphi_{j,\ell}(\mathbf{y}_\ell^{(j)})))$$
for all $j \in [k]$ and for $\mathbf{y}^{(j)}$ as above, where g is the identity function.
- **Reconstruction.** Define
$$\mathsf{Rec}'(\mathbf{z}_1, \dots, \mathbf{z}_k) = \mathsf{Rec}(\mathbf{z}_1, \dots, \mathbf{z}_k).$$

To see why this is correct, notice that by construction, $\{\varphi_{j,r}(\mathbf{y}_r^{(j)}) : j \in [k]\}$ are dt-CNF shares of the secret $f_r(x_{1,r}, \dots, x_{m,r})$. Thus, again by construction, the shares $\mathbf{z}^{(j)}$ are shares under some ℓ-LMSSS with reconstruction algorithm Rec for the concatenated secrets $(f_1(\mathbf{x}_1), \dots, f_\ell(\mathbf{x}_\ell))$. Therefore, $\mathsf{Rec}'(\mathbf{z}_1, \dots, \mathbf{z}_k)$ indeed returns $(f_1(\mathbf{x}_1), \dots, f_\ell(\mathbf{x}_\ell))$, the desired outcome. □

Below are some instantiations of Theorem 3 that yield an HSS with high rate. Example 2 gives an HSS with optimal rate but requires sufficiently large ℓ, while Example 3 has worse rate (and only works for $d = 2$), but uses a significantly smaller ℓ.

Example 2. *Let ℓ, t, k, d, m be integers, and let $\mathbb{F}$ be a finite field. There is a t-private, k-server linear HSS for $\mathsf{POLY}_{d,m}(\mathbb{F})^\ell$, where Share is t-private CNF over $\mathbb{F}$, with download rate at least $1 - dt/k$, for any ℓ of the form $\ell = b(k - dt)$ where $b \geq \log_{|\mathbb{F}|}(k)$ is an integer.*

Proof. We begin with an observation about the existence of linear codes with good rate/distance trade-offs (in particular, that meet the *Singleton bound*).

Fact 1. *Let k, b be positive integers, and suppose that $\mathbb{F}$ is a finite field with $|\mathbb{F}| = q$. Suppose that $q^b \geq k$. For any $R \in (0,1)$ so that $kR \in \mathbb{Z}$, there is an $\mathbb{F}$-linear code $C \subset (\mathbb{F}^b)^k$ with rate R and distance $k(1 - R) + 1$.*

Indeed, we may simply take the standard Reed-Solomon code over $\mathbb{F}_{q^b}$, and then concatenate with the identity code over $\mathbb{F}$, replacing each field element in $\mathbb{F}_{q^b}$ with an appropriate vector in $\mathbb{F}_q^b$.

To obtain the result, combine Theorem 3 with the concatenated Reed-Solomon code of Fact 1. We choose a code of rate $R = 1 - dt/k$ and distance $dt + 1$, and we choose $b \geq \log_{|\mathbb{F}|}(k)$ to be any parameter. As in the theorem statement, we suppose that $\ell = bkR = b(k - dt)$, so that the download rate of the resulting HSS is $\ell/(bk) = R$. □

Example 3. *Let ℓ, r, m be integers, and let $\mathbb{F}$ be a finite field with $|\mathbb{F}| = q$. There is a 1-private, $(k = (q^r - 1)/(q - 1))$-server linear HSS for $\mathsf{POLY}_{2,m}(\mathbb{F})^\ell$, where Share is 1-private CNF over $\mathbb{F}$, with download rate at least $1 - r/k = 1 - O(\log_q(k)/k)$, for $\ell = k - r = k - O(\log_q(k))$.*

Proof. We begin with the following fact, which is achieved by Hamming codes.

Fact 2. *Let r be a positive integer, suppose that $\mathbb{F}$ is a finite field with $|\mathbb{F}| = q$, and set $k = (q^r - 1)/(q - 1)$. There is an $\mathbb{F}$-linear code $C \subset (\mathbb{F})^k$ with dimension $k - r$, rate $1 - r/k$, and distance 3.*

Combining Theorem 3 with Fact 2 yields the desired result, with $\ell = k - r$ and download rate $\ell/k = 1 - r/k$. □

3.1.2 Linear HSS from Shamir Sharing

Our next construction of linear HSS is based on Shamir sharing (Definition 7). Later in Section 3.1.3, we compare this to the CNF-based construction from the previous section.

As noted in Section 1.3, HSS from Shamir sharing is related to work on the use of Reed-Solomon codes as regenerating codes. In particular, existing results in the regenerating codes literature immediately imply the following corollary.

Corollary 2 (Follows from [TYB17]). *Let k, d, m, t be integers so that $k > dt$, and let p be prime. There is a field $\mathbb{F}$ of characteristic p and of size $|\mathbb{F}| = \exp_p(\exp(k \log k))$ so that the following holds. There is a t-private, k-server HSS for $\mathsf{POLY}_{d,m}(\mathbb{F})$ where Share is given by a Shamir scheme (Definition 7, where both the secrets and the shares lie in $\mathbb{F}$) and where the download rate is $1-dt/k$.*

Remark 3 (Limitations of regenerating-code based HSS). *While Corollary 2 has optimal download rate, the fact that the field $\mathbb{F}$ must be so large makes it impractical. On the one hand, if we take the corollary as written, so that $\ell = 1$, each secret must lie in an extremely large extension field, with extension degree exponential in k. On the other hand, it is possible to "pack" $\ell = \exp(k \log k)$ secrets over $\mathbb{F}_p$ into a single element of the large field $\mathbb{F}$; we do this in our Theorem 5, for example. However, the problem with this when starting with a regenerating code is that each secret must be shared independently over the large field $\mathbb{F}$. This means that the upload cost will scale like ℓ^2, while the number of secrets is ℓ. Since $\ell = \exp(k \log k)$ is so large, this results in a huge overhead in upload cost.*

Moreover, [TYB17] shows that a field size nearly this large is necessary for any optimal-download linear repair scheme for Reed-Solomon codes. Thus, we cannot hope to obtain a practical Shamir-based HSS with optimal download rate in a black-box way from regenerating codes.

In light of Remark 3, rather than using an off-the-shelf regenerating code, we adapt ideas from the regenerating codes literature to our purpose. In particular, our approach is inspired by [GW16, TYB17] (and is in fact much simpler). Mirroring our approach for CNF-based sharing, our main tool is an HSS-for-concatenation scheme that begins with Shamir sharing.

Theorem 4 (HSS-for-concatenation from Shamir). *Let $t < k$ be any integers and let $\ell = k - t$. Let $\mathbb{F}$ be any finite field with $|\mathbb{F}| \geq k$ and let $\mathbb{E}$ be an extension field of $\mathbb{F}$ of size $|\mathbb{F}|^\ell$. There is an LSSS $\mathcal{L}$ given by t-private k-party Shamir secret sharing with secrets in $\mathbb{F}$ and shares in $\mathbb{E}$ (as in Definition 7) so that the following holds.*

There is a t-private, k-party, linear (over $\mathbb{F}$) HSS $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ for $\mathsf{CONCAT}_\ell(\mathbb{F})$ so that Share is given by $\mathcal{L}$, and the outputs of Eval are elements of $\mathbb{F}$. In particular, the download rate is $1 - t/k$.

Proof. Let $\mathbb{E}$ be an extension field of $\mathbb{F}$ so that $|\mathbb{E}| = |\mathbb{F}|^\ell$. Let γ be a primitive element of $\mathbb{E}$ over $\mathbb{F}$. We choose $\mathcal{L}$ to be the Shamir scheme with secrets in $\mathbb{F}$, shares in $\mathbb{E}$, and evaluation points $\alpha_0 = \gamma$ and any $\alpha_1, \ldots, \alpha_k \in \mathbb{F}$, using the notation of Definition 7.

Given $\mathcal{L}$, we define Share, Eval, and Rec below.

- **Sharing**. The share function for Π is the same as that for $\mathcal{L}$, and is thus given by $\mathsf{Share}(x_i)_j = p_i(\alpha_j)$, where $p_i \in \mathbb{E}[X]$ is a random polynomial of degree at most t, so that $p_i(\alpha_0) = x_i$. Let $\mathbf{z}^{(j)} := (p_1(\alpha_j), \ldots, p_\ell(\alpha_j))$.

- **Evaluation**. First, we bundle together the ℓ shares to form one Shamir sharing over the larger field $\mathbb{E}$. In more detail, let $y = \sum_{i=1}^{\ell} x_i \gamma^i \in \mathbb{E}$, and define $f(X) \in \mathbb{E}$ by

  $$f(X) = \sum_{i=1}^{\ell} p_i(X)\gamma^i.$$

  Note that $f(X)$ is a degree-t polynomial in $\mathbb{E}[X]$, and that

  $$f(\gamma) = \sum_{i=1}^{\ell} p_i(\gamma)\gamma^i = \sum_{i=1}^{\ell} x_i\gamma^i = y.$$

  Thus, if each server j locally computes $z^{(j)} = f(\alpha_j) \in \mathbb{E}$ from its shares, they now have a valid Shamir sharing over $\mathbb{E}$ of the secret $y \in \mathbb{E}$.

  Since $\ell = k - t$, for any polynomial $f \in \mathbb{E}[X]$ of degree at most t, and any $r \in [\ell]$, we have $\deg(X^{r-1} \cdot f(X)) \leq (k-t-1)+t = k-1$. Thus, for each $r \in [\ell]$, there is a linear relationship between the $k+1$ evaluations $\alpha_1^{r-1}f(\alpha_1), \ldots, \alpha_k^{r-1}f(\alpha_k)$ and $\gamma^{r-1}f(\gamma)$, in the sense that there are some coefficients $\lambda_i \in \mathbb{E}$ (independent of f and r) so that for any $f \in \mathbb{E}[X]$ of degree at most t, and any $r \in [\ell]$,

  $$\gamma^{r-1}f(\gamma) = \sum_{i=1}^{k} \lambda_i \alpha_i^{r-1} f(\alpha_i).$$

  Define

  $$w^{(j)} = \mathrm{tr}(\lambda_j f(\alpha_j)),$$

  where $\mathrm{tr}(\cdot)$ denotes the *field trace* of $\mathbb{E}$ over $\mathbb{F}$, defined as $\mathrm{tr}(X) = \sum_{i=0}^{\ell-1} X^{|\mathbb{F}|^i}$. We define

  $$\mathsf{Eval}(f, j, \mathbf{z}^{(j)}) = w^{(j)}.$$

  Notice that $w^{(j)} \in \mathbb{F}$, since the image of tr is $\mathbb{F}$.

- **Reconstruction**. Suppose we are given $w^{(1)}, \ldots, w^{(k)}$ produced by Eval. Our goal is to recover $(x_1, \ldots, x_\ell) \in \mathbb{F}^\ell$ in an $\mathbb{F}$-linear way.

  Since tr is $\mathbb{F}$-linear and $\alpha_j \in \mathbb{F}$ for all $j \in [k]$, we have for all $r \in [\ell]$ that

  $$\begin{aligned}\mathrm{tr}(\gamma^{r-1}f(\gamma)) &= \mathrm{tr}\left(\sum_{j=1}^{k} \lambda_j \alpha_j^{r-1} f(\alpha_j)\right) \\ &= \sum_{j=1}^{k} \alpha_j^{r-1}\mathrm{tr}(\lambda_j f(\alpha_j)) \\ &= \sum_{j=1}^{k} \alpha_j^{r-1} w^{(j)}\end{aligned}$$

  Therefore, there is an $\mathbb{F}$-linear map to recover each $\mathrm{tr}(\gamma^{r-1}f(\gamma))$ for $r \in [\ell]$, given the shares $w^{(j)}$. Since $\gamma^0, \ldots, \gamma^{\ell-1}$ form a basis for $\mathbb{E}$ over $\mathbb{F}$, there is also an $\mathbb{F}$-linear function that recovers $f(\gamma) = y$ from $\mathrm{tr}(\gamma^{r-1}f(\gamma))$ for $r \in [\ell]$. Finally, since y is defined as $y = \sum_{j=1}^{\ell} x_i\gamma^i$

and $\gamma, \gamma^2, \ldots, \gamma^\ell$ form a basis for $\mathbb{E}$ over $\mathbb{F}$, there is an $\mathbb{F}$-linear map to recover the coefficients x_i from y. Composing all of these, we see that there is an $\mathbb{F}$-linear map that can recover $(x_1, \ldots, x_\ell)$ given the shares $w^{(1)}, \ldots, w^{(k)}$. This map will be Rec.

Finally, we observe that the information rate of $\mathcal{L}$ is $\frac{\ell}{k} = 1 - t/k$, as there are $\ell = t - k$ elements of $\mathbb{F}$ that are shared as k elements of $\mathbb{F}$. $\square$

In order to extend Theorem 4 to an HSS for POLY_d^ℓ, we will use the well-known local multiplication property of Shamir sharing (e.g., [BGW88, CCD88, CDM00]).

Fact 3. *Let $\mathbb{F} \leq \mathbb{E}$ be finite fields and let $\alpha_0, \ldots, \alpha_k \in \mathbb{E}$ be distinct. Let $\mathcal{L}$ be the d-LMSSS that is given by d instances of the t-private, k-party Shamir sharing scheme with secrets in $\mathbb{F}$, shares in $\mathbb{E}$, and evaluation points $(\alpha_0, \ldots, \alpha_k)$ (Definition 7). Let $\mathcal{L}'$ be the LSSS given by the dt-private, k-party Shamir sharing scheme with the same specifications. Let $\psi(x_1, \ldots, x_d) = \prod_{i=1}^d x_i$. Then $\mathcal{L}$ is locally convertible to $\mathcal{L}'$ with respect to ψ.*

Remark 4 (Comparison of Theorem 4 to the straightforward scheme)**.** *As noted in Remark 2, there is a straightforward HSS-for-concatenation with a different version of Shamir sharing, which arises from the ℓ-LMSSS of Franklin-Yung [FY92] (Remark 1(b)). This scheme matches the parameters of Theorem 4 (and in particular is optimal for the parameters to which it applies). However, we cannot use that straightforward scheme for our next step, which is to generalize to POLY_d^ℓ. The reason is that the generalization uses Fact 3. This fact is not true for the sharing that arises from Remark 2, since to ensure t-privacy, the degree of the polynomials used to share must be $t + \ell - 1$, rather than t.*

By combining Fact 3 with Theorem 4 we conclude the following.

Theorem 5 (HSS-for-polynomials from Shamir)**.** *Let $\mathbb{F}$ be a finite field. Let m be a positive integer. Let $b \geq \log_{|\mathbb{F}|}(k)$ be a positive integer and let $\ell = b(k - dt)$. There is a t-private k-server linear HSS $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ for $\mathsf{POLY}_{d,m}(\mathbb{F})^\ell$ so that Share is Shamir sharing (Definition 7), where the upload cost is $kmb^2(k - dt)^2 \log_2 |\mathbb{F}|$, and the download cost is $kb \log_2 |\mathbb{F}|$. Consequently, the download rate is $1 - dt/k$.*

Proof. We first observe that a proof nearly identical to the proof of the first part of Theorem 3 establishes the theorem for $b = 1$, if it holds that $|\mathbb{F}| \geq k$. (Indeed, we use Theorem 4 rather than Theorem 1 for the HSS-for-concatenation, and Fact 3 rather than Lemma 2 for the share conversion.) To see the upload and download costs for the $b = 1$ case, notice that each server $j \in [k]$ holds as an input share an element of $\mathbb{E}$ (as in the statement of Theorem 4), which is $\log_2 |\mathbb{E}| = \ell \log_2 |\mathbb{F}|$ bits. Thus the upload cost is $km\ell^2 \log_2 |\mathbb{F}| = km(k - dt)^2 \log_2 |\mathbb{F}|$ bits. Each output share is a single element of $\mathbb{F}$, so the download cost is $k \log_2 |\mathbb{F}|$. Thus, the download rate is

$$\frac{\ell \log_2 |\mathbb{F}|}{k \log_2 |\mathbb{F}|} = 1 - dt/k,$$

as desired.

In order to prove the theorem for $b \geq 1$ (and hence possibly for $|\mathbb{F}| < k$), let $\tilde{\mathbb{F}}$ be an extension field of $\mathbb{F}$ of degree b, so that $|\tilde{\mathbb{F}}| = |\mathbb{F}|^b \geq k$. Let γ be a primitive element of $\tilde{\mathbb{F}}$ over $\mathbb{F}$. Then let $\Pi' = (\mathsf{Share}', \mathsf{Eval}', \mathsf{Rec}')$ be the HSS scheme over $\tilde{\mathbb{F}}$ guaranteed by the $b = 1$ case by the above, for $\tilde{\ell} = k - dt$. We will define $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ as follows.

- **Sharing.** Suppose the inputs to Π are $\mathbf{x}_1, \dots, \mathbf{x}_\ell \in \mathbb{F}^m$. Organize these inputs as $\mathbf{x}_{i,r}$ for $i \in [b]$ and $r \in [\tilde{\ell}]$. The Share function shares each $x_{i,r}$ independently by applying Share' to $(x_{i,r})_{r\in[\tilde{\ell}]}$ independently for each $i \in [b]$. Suppose that these input shares are given by $y_{i,r}^{(j)} \in \tilde{\mathbb{F}}$ for each $i \in [b], r \in [\tilde{\ell}]$ and for each server $j \in [k]$.

- **Evaluation.** For $f \in \mathsf{POLY}_{d,m}(\mathbb{F})^\ell$ given by $(f_1, \dots, f_\ell)$ and re-organized as $(f_{i,r})_{i\in[b],r\in[\tilde{\ell}]}$, define
$$z^{(j)} := \mathsf{Eval}(f, j, (y_{i,r}^{(j)})_{i\in[b],r\in[\tilde{\ell}]}) = \sum_{i\in[b]} \mathsf{Eval}'(f, j, (y_{i,r}^{(j)})_{r\in[\tilde{\ell}]})\gamma^i,$$
so that $z^{(j)} \in \tilde{\mathbb{F}}$.

- **Reconstruction.** Suppose the output shares are $z_1, \dots, z_k$ as above. Then by linearity and the correctness of Rec',
$$\begin{aligned}\mathsf{Rec}'(z^{(1)}, \dots, z^{(k)}) &= \sum_{i\in[b]} \gamma^i \mathsf{Rec}'(\mathsf{Eval}'(f, j, ((y_{i,r}^{(j)})_{r\in[\tilde{\ell}]})_{j\in[k]})) \\ &= \sum_{i\in[b]} \gamma^i (f_{i,r}(x_{i,r}))_{r\in[\tilde{\ell}]}.\end{aligned}$$
Since $\gamma, \gamma^2, \dots, \gamma^b$ form a basis of $\tilde{\mathbb{F}}$ over $\mathbb{F}$, and since $(f_{i,r}(x_{i,r})_{r\in[\tilde{\ell}]} \in \mathbb{F}^{\tilde{\ell}}$, given this, we can recover all of the vectors $(f(x_{i,r})_{r\in[\tilde{\ell}]} \in \mathbb{F}^{\tilde{\ell}}$ for all $i \in [b]$. Thus we, after re-arranging, we have assembled $(f_s(x_s))_{s\in[\ell]}$, as desired. This defines Rec.

It remains to verify the upload and download costs. The upload cost is b times the upload cost for Π', for a total of
$$b \cdot km(k - dt)^2 \log_2 |\tilde{\mathbb{F}}| = b^2 km(k - dt)^2 \log_2 |\mathbb{F}|.$$
The download cost is the same as the download cost for Π', which is
$$k \log_2 |\tilde{\mathbb{F}}| = kb \log_2 |\mathbb{F}|.$$
Consequently, the download rate is
$$\frac{\ell \log_2 |\mathbb{F}|}{kb \log_2 |\mathbb{F}|} = 1 - dt/k.$$
□

Finally, note that if b is a multiple of $\lceil \log_{|\mathbb{F}|}(k) \rceil$ in Theorem 5, then the upload cost can be improved to $kmb\lceil \log_{|\mathbb{F}|}(k)\rceil(k - dt)^2 \log_2 |\mathbb{F}|$. This is achieved by initializing the scheme from Theorem 5 with $b = \lceil \log_{|\mathbb{F}|}(k)\rceil$, and then repeating it $b/\lceil \log_{|\mathbb{F}|}(k)\rceil$ times.

3.1.3 Comparison between CNF-based HSS and Shamir-based HSS

In this section we comment briefly on the differences between Theorem 3 and Theorem 5 in particular and on the differences between CNF and Shamir sharing more generally.

Comparison of our HSS schemes. For simplicity, we focus on our schemes for HSS-for-concatenation (Theorems 2 and 4), and the case where $m = 1$ and where the Eval functions of both schemes output a single element of the same field (that is, $b = 1$). Thus, there are only two parameters to adjust: the number of parties k and the privacy t. The goal is to maximize the download rate (or equivalently to minimize the number of repetitions ℓ).

Note that for our Shamir-based HSS, the download rate is $1 - t/k$ whenever the scheme is defined, and in particular when $b = 1$ we always take $\ell = k - t$ and enforce $|\mathbb{F}| \geq k$. Note that our CNF-based HSS scheme is defined for any field and any ℓ, since linear codes exist for all such choices.

Both our CNF-based HSS and our Shamir-based HSS achieve the optimal download rate $1-t/k$ when they are both defined, but there are three important differences.

- **Upload cost.** The main advantage of the Shamir-based HSS (Theorem 4) is that the upload cost is smaller: when $\ell = k-t$ (so that our Shamir-based HSS exists), the upload cost for our Shamir-based HSS is $k(k-t)^2 \log_2 |\mathbb{F}|$ bits, while the upload cost for our CNF-based HSS is $k(k-t)\binom{k-1}{t} \log_2 |\mathbb{F}|$ bits. Thus, the Shamir-based HSS has a smaller upload cost whenever $k - t < \binom{k-1}{t}$.

- **Flexibility.** The main advantage of our CNF-based HSS is that the parameter regime in which it is defined is much less restrictive. More concretely, the Shamir-based HSS requires that the field $\mathbb{F}$ have size at least k (when $b = 1$ as we consider here), and it requires $\ell \geq k-t$. In contrast, the CNF-based HSS applies for any ℓ and over any field, as linear codes of rate $\ell/(kb)$ exist for any ℓ over any field.

- **Download rate.** As mentioned above, whenever the Shamir-based HSS is defined, both schemes have the same (optimal) download rate. However, the CNF-based HSS applies more generally, and in the case of HSS-for-concatenation has optimal rate for any linear HSS whenever it is defined, as per Theorem 2.

  In order to get a meaningful comparison between our two HSS constructions, one can try to apply the Shamir-based HSS over a field of size less than k by embedding the secrets into a field $\mathbb{K} \geq \mathbb{F}$, where $|\mathbb{K}| \geq k$. When we do this, the CNF-based HSS attains a strictly better download rate.

  For example, suppose that $\mathbb{F} = \mathbb{F}_2$, and $t = 2$, and suppose for simplicity that $k = 2^r - 1$ for some r. Furthermore, suppose that we only allow $\ell \leq k$. In this case, the best CNF-based HSS is given by the best binary linear code with distance 3 and length $2^r - 1$, as per Theorem 2. This is the Hamming code of length $2^r - 1$ (see Fact 2). Translating the parameters of the Hamming code to HSS, the CNF-based HSS has $\ell = k - r$ and hence download rate

  $$1 - \frac{r}{k} \approx 1 - \frac{\log k}{k}.$$

  In contrast, the Shamir-based HSS has $\ell = k - 2$ (recalling that $t = 2$ for this comparison), but is working over a bigger field of size 2^r and so has download rate

  $$\frac{1 - t/k}{r} \approx \frac{1 - 2/k}{\log k}.$$

Comparison of CNF and Shamir sharing, with any HSS scheme. One may wonder whether there is an inherent limitation to Shamir sharing relative to CNF sharing. That is, could it be possible that *some* HSS based on Shamir sharing can match the performance of our CNF-based sharing scheme?

More precisely, we ask if there is a Shamir-based HSS that can match the performance of our CNF-based HSS, *where the field* $\mathbb{E}$ *that the Shamir shares lie in is the smallest such that* $|\mathbb{E}| > k$ (aka, the smallest so that Shamir sharing is defined). The reason to phrase the question this way is that if we allowed $\mathbb{E}$ to be arbitrarily large, it might be the case that the Shamir-based HSS had a larger share size than the CNF-based HSS, undoing the advantage of the Shamir-based HSS.

It turns out that the answer to this question can be "no," as the following example shows.

Proposition 1. *Consider the problem of constructing* 1*-private* 5*-server HSS for* $\mathsf{CONCAT}_4(\mathbb{F}_2)$.

There is a $\mathbb{F}_2$*-linear HSS that solves this problem with CNF sharing and download cost* 5 *bits (that is, one bit per server). However, there is* no $\mathbb{F}_2$*-linear HSS that solves this problem with Shamir sharing and with download cost* 5*, where each of the* 5 *secrets in* $\mathbb{F} = \mathbb{F}_2$ *are shared independently over* $\mathbb{E} = \mathbb{F}_8$.

We defer the proof of Proposition 1 to Appendix B. Briefly, the possibility result (that is, that there is such a CNF-based HSS) follows from an example of such a scheme. The impossibility result (that is, that there is no such Shamir-based HSS) follows from a computer search. Naively such a search (for example, over all sets of $k = 5$ linear functions from $\mathbb{F}_8^4 \to \mathbb{F}_2$) is not computationally tractable. Instead we first analytically reduce the problem to one that is tractable, and then run the search.

Notice that the share size per server is 16 bits with CNF, and 12 bits with Shamir (when the shares are in $\mathbb{F}_8$). In particular, if $\mathbb{E}$ were any larger extension field of $\mathbb{F}_2$ than $\mathbb{F}_8$, the Shamir scheme an upload cost at least as large as the CNF scheme. Thus, the example when $\mathbb{E} = \mathbb{F}_8$ is the most generous for Shamir-based HSS if we demand that the Shamir-based HSS still has an upload cost advantage.

3.1.4 Application to Private Information Retrieval

As a simple application of HSS for low-degree polynomials, we extend the information-theoretic PIR protocol from [CGKS98] to allow better download rate by employing more servers, while maintaining the same (sublinear) upload cost.

Theorem 6 (PIR with sublinear upload cost and high download rate)**.** *For all integers* $d, t, k, w > 0$*, such that* $dt + 1 \le k$*, there is a* t*-private* k*-server PIR protocol for* $(w \cdot (k - dt) \cdot \lceil \log_2 k \rceil)$*-bit record databases of size* N *such that:*

- *The upload cost is* $O(k^3 \log k \cdot N^{1/d})$ *bits;*
- *The download cost is* $wk\lceil \log_2 k \rceil$ *bits. Consequently, the rate of the PIR is* $1 - dt/k$.

Proof. Without loss of generality, assume $w = 1$, as the scheme can be repeated to apply for arbitrary w. To this end, we view the database of size N as $k - dt$ vectors $(D_{i,1}, \ldots, D_{i,N}) \in (\mathbb{F}_{2^{\lceil \log_2 k \rceil}})^N$, $i = 1, \ldots, k - dt$, such that each record with index $j \in [N]$ in the database consists of $(D_{1,j}, D_{2,j}, \ldots, D_{k-dt,j})$. Next, fix an injective mapping $\eta : [N] \to W(d, m)$, where $W(d, m)$ is the set of all vectors from $\{0, 1\}^m$ with Hamming weight d, $m > 0$ is the smallest integer such that

$N \leq \binom{m}{d}$. In addition, for $j \in [N]$ and $\ell \in [d]$ let $\eta(j)_\ell$ denote the ℓ'th nonzero coordinate of $\eta(j)$. Furthermore, for $i = 1, \ldots, k - dt$ define the polynomial over $\mathbb{F}_{2^{\lceil \log_2 k \rceil}}[z_1, \ldots, z_m]$:

$$p_i(z_1, \ldots, z_m) = \sum_{j=1}^{N} D_{i,j} \cdot z_{\eta(j)_1} \cdots z_{\eta(j)_d}$$

Now, notice that the polynomials $p_1, \ldots, p_{k-dt}$ jointly encode the database, in the sense that for every $j \in [N]$, $i = 1, \ldots, k - dt$ we have $p_i(\eta(j)) = D_{i,j}$.

Let Π be the t-private k-server HSS from Theorem 5 for $\mathsf{POLY}_{d,m}(\mathbb{F}_{2^{\lceil \log_2 k \rceil}})^{(k-dt)}$ (with the choice of $b = 1$). Suppose that the client wishes to recover the j'th record $(D_{1,j}, D_{2,j}, \ldots, D_{k-dt,j})$. By construction, the client needs to receive the evaluation of all $k - dt$ polynomials p_i on point $\eta(j)$. Thus, the client uses Π to share the $\eta(j)$ among the k servers; t-privacy of the PIR protocol follows from t-privacy of Π. Next, the servers each compute their output shares according to Π and return them to the client. The client runs Π's reconstruction algorithm to recover $p_i(\eta(j))$ for each i. This determines the j'th record as noted above.

By the guarantees of Theorem 5, the upload cost is $k \cdot m \cdot (k - dt)^2 \cdot \lceil \log_2 k \rceil = O(k^3 \log k \cdot N^{1/d})$, as $m = O(N^{1/d})$. In addition, the download cost is $wk\lceil \log_2 k \rceil$ bits. $\square$

3.2 Negative Results for Linear HSS

In Section 3.1.1, we obtained linear HSS schemes for $\mathsf{POLY}_{d,m}(\mathbb{F})^\ell$ with rate $1 - dt/k$. Our main result in this section, Theorem 7, shows that this bound is tight for *linear* HSS. In particular, this implies that for linear HSS, there is an inherent overhead for polynomials of larger degree in terms of download cost or rate.

Moreover, we also provide some simple negative results that (a) strengthen the bound of Theorem 7 for linear HSS-for-concatenation for specific parameters (Proposition 3), and (b) provide a negative result for general (not necessarily linear) HSS, which is tight for $d = 1$ (Theorem 3).

Theorem 7. *Let t, k, d, m, ℓ be positive integers so that $m \geq d$. Let $\mathbb{F}$ be any finite field. Let $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ be a t-private k-server linear HSS for $\mathsf{POLY}_{d,m}(\mathbb{F})^\ell$. Then $dt < k$, and the download cost of Π is at least $k\ell \log_2 |\mathbb{F}|/(k - dt)$. Consequently, the download rate of Π is at most $1 - dt/k$.*

Remark 5. *An inspection of the proof shows that Theorem 7 still holds if Rec is allowed to be linear over some subfield $\mathbb{B}$ of $\mathbb{F}$, rather than over $\mathbb{F}$ itself.*

Proof of Theorem 7. Suppose that $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ is a t-private k-server linear HSS scheme for POLY_d^ℓ. By the definition of linearity (Definition 3) and of POLY_d^ℓ (Definition 9), we have $\mathcal{X} = \mathcal{Y} = \mathbb{F}$ for some finite field $\mathbb{F}$; $\mathsf{Share}(\cdot)$ is linear over $\mathbb{F}$; each output share $\mathbf{y}^{(j)}$ (output of Eval) is an element of $\mathbb{F}^{b_j}$ for some $b_j \geq 0$; and $\mathsf{Rec} : \mathbb{F}^{\sum_j b_j} \to \mathbb{F}^\ell$ is linear over $\mathbb{F}$. Suppose without loss of generality that $b_1 \leq b_2 \leq \cdots \leq b_k$.

Without loss of generality, we may assume that Share is t-CNF sharing; indeed, by Theorem 1, t-CNF shares can be locally converted to shares of any t-private secret-sharing scheme with a linear Share function.

Also without loss of generality, it suffices to show an impossibility result for the special case where there are $d\ell$ inputs, and the HSS is for the class $\mathcal{F}^\ell$, where $\mathcal{F}$ consists only of the monomial $f(x_1, \ldots, x_d) = \prod_{i=1}^d x_i$. Indeed, this is because $\mathcal{F} \subset \mathsf{POLY}_d$.

Suppose that the secrets are $x_{i,r}$ for $i \in [d]$ and for $r \in [\ell]$, so that we want to compute $\prod_{i=1}^d x_{i,r}$ for each $r \in [\ell]$. In order to share them according to t-CNF (Definition 6), we choose random $X_{i,r}^S \in \mathbb{F}$ for all $i \in [d], r \in [\ell]$, and all $S \subset [k]$ of size t, so that for all i, r,

$$x_{i,r} = \sum_S X_{i,r}^S, \tag{2}$$

where the sum is over all sets $S \subset [k]$ of size t. Then server j receives the input share $\mathbf{x}^{(j)} = (X_{i,r}^S : j \notin S)$ and generates an output share $\mathbf{y}^{(j)} \in \mathbb{F}^{b_j}$ via the Eval map.

We may treat each output share $\mathbf{y}^{(j)}$ as a vector of polynomials $\mathbf{p}^{(j)}(\mathbf{X}) = (p_1^{(j)}(\mathbf{X}), \ldots, p_{b_j}^{(j)}(\mathbf{X}))$ in the variables $\mathbf{X} = (X_{i,r}^S)_{i\in[d], r\in[\ell], S\subset[k] \text{ with } |S|=t}$. Since Rec is $\mathbb{F}$-linear, there are some vectors $\mathbf{v}_r^{(j)} \in \mathbb{F}^{b_j}$ for each $j \in [k], r \in [\ell]$ so that the r'th output of $\mathsf{Rec}(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)})$ is given by

$$p_r(\mathbf{X}) := \sum_{j=1}^k \langle \mathbf{v}_r^{(j)}, \mathbf{p}^{(j)}(\mathbf{X}) \rangle. \tag{3}$$

By the correctness of Π, this output must be equal to $\prod_{i=1}^d x_{i,r}$. Plugging in (2), this reads

$$p_r(\mathbf{X}) = \sum_{S_1,\ldots,S_d} \left(\prod_{i=1}^d X_{i,r}^{S_i} \right), \tag{4}$$

where the sum is over all choices of $S_1, \ldots, S_d \subset [k]$ of size t. Define ℓ polynomials $w_r(\mathbf{X})$ by

$$w_r(\mathbf{X}) = \sum_{j=1}^{k-dt} \langle \mathbf{v}_r^{(j)}, \mathbf{p}^{(j)}(\mathbf{X}) \rangle$$

for $r \in [\ell]$. We will show below in Claim 1 that, on the one hand, the w_r are linearly independent over $\mathbb{F}$; but on the other hand, they are all clearly contained in a $\left(\sum_{j=1}^{k-dt} b_j\right)$-dimensional subspace, spanned by the polynomials $p_h^{(j)}(\mathbf{X})$ for $j \in [k-dt]$ and $h \in [b_j]$ (recalling that $\mathbf{p}^{(j)} = (p_h^{(j)})_{h\in[b_j]}$). This implies that

$$\ell \leq \sum_{j=1}^{k-dt} b_j \leq \max \left\{ 0, \frac{k-dt}{k} \sum_{j=1}^k b_j \right\},$$

where we have used the assumption that $b_1 \leq b_2 \leq \cdots \leq b_k$. Rearranging (and using the fact that $\ell > 0$), this implies that $dt < k$ and that the scheme downloads at least

$$\sum_{j=1}^k b_j \geq \ell \left(\frac{k}{k-dt} \right)$$

symbols of $\mathbb{F}$, or at least $k\ell \log_2 |\mathbb{F}|/(k-dt)$ bits, which will prove the theorem.

Thus, it suffices to prove the following claim.

Claim 1. *The polynomials $w_r(\mathbf{X})$ are linearly independent over $\mathbb{F}$.*

Proof. Choose sets $S_1^*, S_2^*, \ldots, S_d^* \subseteq [k]$ with $|S_i^*| = t$ so that

$$\bigcup_{i=1}^{d} S_i^* = \{k - dt + 1, k - dt + 2, \ldots, k\}.$$

If $dt < k$, then the S_i^* can be any sets that cover the last dt elements of $[k]$. Then consider the monomial

$$M_r(\mathbf{X}) := \prod_{i=1}^{d} X_{i,r}^{S_i^*}$$

for $r \in [\ell]$. From (4), we see that $M_r(\mathbf{X})$ appears in $p_r(\mathbf{X})$. However, since for all $j > k - dt$ there exists $i \in [d]$ such that $j \in S_i^*$, $M_r(\mathbf{X})$ cannot appear in $\mathbf{p}^{(j)}(\mathbf{X})$ for any $j > k - dt$. Then (3) implies that $M_r(\mathbf{X})$ must appear in $w_r(\mathbf{X})$.

Further, for any $r' \neq r$, $M_r(\mathbf{X})$ cannot appear in $w_{r'}(\mathbf{X})$. Indeed, since $M_r(\mathbf{X})$ does not appear in $p_{r'}(\mathbf{X})$ (from (4)), if $M_r(\mathbf{X})$ appeared in $w_{r'}(\mathbf{X})$, then (3) implies that it must be canceled by a contribution by some $\mathbf{p}^{(j)}(\mathbf{X})$ for $j > k - dt$. But, as above, $M_r(\mathbf{X})$ cannot appear in $\mathbf{p}^{(j)}(\mathbf{X})$.

Thus, each $M_r(\mathbf{X})$ appears in $w_{r'}(\mathbf{X})$ if and only if $r' = r$. Since the monomials $M_r(\mathbf{X})$ are linearly independent over $\mathbb{F}$, this implies that the polynomials $w_r(\mathbf{X})$ are also linearly independent over $\mathbb{F}$. This proves the claim. □

With the claim proved, the theorem follows. □

Theorem 7 implies that any linear HSS scheme for concatenation must have download rate at most $1 - t/k$. As we have seen, this is achievable in some parameter regimes, but it turns out that it is not achievable for all parameter regimes. In particular, in the corollary below, we observe that Theorem 2 immediately rules this out, based on known bounds on the distance of linear codes.

Corollary 3. *Let k, b, t, ℓ be integers so that $\ell < kb - 1$. Suppose that Π is a linear t-private k-server HSS for $\mathsf{CONCAT}_\ell(\mathbb{F})$ with output shares in $\mathbb{F}^b$, and suppose that $|\mathbb{F}|^b < k/2$. Then the download rate of Π is strictly less than $1 - t/k$.*

Proof. It is known (see, e.g., [Bal12]) that any linear code C of length k over an alphabet of size $q < k/2$ with $\dim(C) < k - 1$ *cannot* achieve the Singleton bound; that is, we must have $\dim(C) < k - \mathrm{Dist}(C) + 1$. Thus, Theorem 2 implies the corollary. □

Finally, we note that for general (not necessarily linear) information-theoretic HSS, one can also show a negative result that implies that some degredation in rate is inevitable for larger security thresholds t. However, this result does not depend on the complexity of $\mathcal{F}$, and in fact applies to families $\mathcal{F}$ with only one function in them. The proof below is similar to one that appears in [HLKB16] for low-bandwidth secret sharing.

Lemma 3 (Similar to Prop. 1 in [HLKB16])**.** *Let Π be any t-private k-server HSS for a function class $\mathcal{F}$ with output alphabet $\mathcal{Y}$, so that there is some $f \in \mathcal{F}$ with $\mathrm{Im}(f) = \mathcal{Y}$. Then Π must have download cost at least $\frac{k \log_2 |\mathcal{Y}|}{k-t}$. Consequently, its download rate is at most $1 - t/k$.*

Proof. Fix a function $f \in \mathcal{F}$ so that $\mathrm{Im}(f) = \mathcal{Y}$. Choose a secret $\mathbf{x} \in \mathcal{X}^m$ randomly, according to some distribution so that $f(\mathbf{x})$ is uniformly distributed in $\mathcal{Y}$. (Such a distribution exists since the image of f is $\mathcal{Y}$). Suppose that $y^{(1)}, \ldots, y^{(k)}$ are the output shares (outputs of Eval), so that

$y^{(j)} \in \{0,1\}^{b_j}$ is b_j bits long. Assume without loss of generality that $b_1 \geq b_2 \geq \cdots \geq b_k$. Then we have

$$\begin{aligned} H(\mathbf{x}) &= H(\mathbf{x}|y^{(1)}, \ldots, y^{(t)}) && (5)\\ &\leq H(\mathbf{x}, y^{(t+1)}, \ldots, y^{(k)}|y^{(1)}, \ldots, y^{(t)}) \\ &= H(\mathbf{x}|y^{(1)}, \ldots, y^{(k)}) + H(y^{(t+1)}, \ldots, y^{(k)}|y^{(1)}, \ldots, y^{(t)}) && (6)\\ &\leq H(\mathbf{x}|f(\mathbf{x})) + H(y^{(t+1)}, \ldots, y^{(k)}), && (7) \end{aligned}$$

where (5) follows from t-privacy, (6) follows from the chain rule for conditional entropy, and (7) follows from the facts that $y^{(1)}, \ldots, y^{(k)}$ determine $f(\mathbf{x})$ and that $H(A|B) \leq H(A|g(B))$ for any function g and any random variables A, B. Rearranging and using the fact that $H(y^{(t+1)}, \ldots, y^{(k)}) \leq b^{(t+1)} + \ldots + b^{(k)}$, we conclude that

$$H(\mathbf{x}) - H(\mathbf{x}|f(\mathbf{x})) \leq H(y^{(t+1)}, \ldots, y^{(k)}) \leq b_{t+1} + \cdots + b_k \leq \frac{k-t}{k}(b_1 + \cdots + b_k),$$

using the assumption that $b_1 \geq b_2 \geq \cdots \geq b_k$. Finally, we have

$$H(\mathbf{x}) - H(\mathbf{x}|f(\mathbf{x})) = I(\mathbf{x}; f(\mathbf{x})) = H(f(\mathbf{x})) - H(f(\mathbf{x})|\mathbf{x}) = H(f(\mathbf{x})),$$

because since f is fixed, $H(f(\mathbf{x})|\mathbf{x}) = 0$. We conclude that

$$H(f(\mathbf{x})) \leq \frac{k-t}{k}(b_1 + \cdots + b_k).$$

Since $f(\mathbf{x})$ is uniform in $\mathcal{Y}$, we have $H(f(\mathbf{x})) = \log_2 |\mathcal{Y}|$, and re-arranging we see that

$$b_1 + \cdots + b_k \geq \frac{k \log_2 |\mathcal{Y}|}{k-t},$$

as desired. □

4 Black-Box Rate Amplification

In this section we present a *generic* rate amplification technique for HSS. More concretely, we show how to make a black-box use of any t_0-private k_0-server HSS scheme $\Pi_0 = (\mathsf{Share}_0, \mathsf{Eval}_0, \mathsf{Rec}_0)$ for $\mathcal{F}$ with *additive reconstruction* (see Definition 3) to obtain a t-private k-server HSS scheme $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ (where $t/k < t_0/k_0$) for $\mathcal{F}^\ell$ with better download rate and similar upload cost. This construction can be applied, with useful corollaries, to both information-theoretic and computational HSS.

After laying out the general approach (Lemma 4), we present the three useful instances of this technique discussed in the introduction. First, in Section 4.1 we obtain a general-purpose (computationally) t-private k-server HSS Π with rate $1 - t/k$. Second, in Section 4.2, we show how to transform any 1-private 2-server HSS scheme Π_0 into a 1-private k-server HSS scheme Π with rate $1 - 1/k$, and obtain high-rate computationally secure PIR and HSS. Finally, in Section 4.3, we obtain (information theoretic) PIR with sub-polynomial ($N^{o(1)}$) upload cost and download rate approaching 1.

The general approach. Let $\Pi_0 = (\mathsf{Share}_0, \mathsf{Eval}_0, \mathsf{Rec}_0)$ be a t_0-private, k_0-server HSS for $\mathcal{F}$ *with additive reconstruction over* $\mathbb{F}$. We will construct a t-private, k-server HSS Π for $\mathcal{F}^\ell$, using Π_0 as a black box. Suppose that we have inputs $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ and we wish to compute $(f_1(\mathbf{x}_1), \ldots, f_\ell(\mathbf{x}_\ell))$, for $f_i \in \mathcal{F}$. The general paradigm proceeds as follows:

1. Using Share_0, we share the inputs as input shares $\mathbf{x}_i^{(v)}$, for $i \in [\ell], v \in [k_0]$. In order to define Share in Π, these input shares $\mathbf{x}_i^{(v)}$ are replicated, *somehow*, between the k servers.

2. Each server $j \in [k]$ applies Eval_0 to the input shares $\mathbf{x}_i^{(v)}$ that they hold, to obtain output shares $y_i^{(v)}$. Thus, $(y_i^{(v)})_{v\in[k_0]}$ are additive shares of $f_i(\mathbf{x}_i)$: $\sum_{v\in[k_0]} y_i^{(v)} = f_i(\mathbf{x}_i)$. Now each server j holds some collection $\mathbf{Y}(j)$ of the $y_i^{(v)}$. These $\mathbf{Y}(j)$ form shares of $(f_1(\mathbf{x}_1), \ldots, f_\ell(\mathbf{x}_\ell))$ under some ℓ-LMSSS $\mathcal{L}_0$.

   In order to define Eval for Π, we use *some* share conversion from $\mathcal{L}_0$ to a new ℓ-LMSSS $\mathcal{L}$. The download rate of Π is thus given by the information rate of $\mathcal{L}$.

3. The output client uses the reconstruction algorithm for $\mathcal{L}$ in order to define Rec for Π.

This approach allows essentially two degrees of freedom: First, in Step 1 we may choose how to replicate the shares; and second, in Step 2 we may choose the share conversion and the new LMSSS $\mathcal{L}$. Hence, since the shares $y_i^{(v)}$ are k_0-additive shares, the above can be seen as a *generalized CNF share conversion question* à la Theorem 1: instead of using the usual CNF sharing from Definition 6, each secret is additively shared into only k_0 shares, and there is the freedom to choose which server receives which share. In general, for all integers t, k, t_0, k_0, ℓ, we will be interested in whether a t-private k-server HSS Π for $\mathcal{F}^\ell$ can be constructed from a t_0-private k_0-server HSS Π_0 for $\mathcal{F}$ using the black-box approach above. As we will see later, for our purposes it will be sufficient to restrict ourselves to $t_0 = (k_0 - 1)$-private k_0-server schemes, as even for $(t_0 < k_0 - 1)$-private Π_0, we could still apply black-box transformations which preserve $k_0 - 1$ privacy (see Remark 6). We can capture this approach with the following notion:

Definition 11 (Black-box transformation)**.** *Let t, k, k_0, ℓ be integers, and $\mathbb{F}$ a finite field. We say there is a* black-box transformation for $(t, k, k_0, \ell, \mathbb{F})$, *if there is k-server ℓ-LMSSS over $\mathbb{F}$ $\mathcal{L} = (\mathsf{Share}_\mathcal{L}, \mathsf{Rec}_\mathcal{L})$ with parameters $(e, b_1, \ldots, b_k)$,* replication functions $\psi_i : [k_0] \to 2^{[k]}$*, $i \in [\ell]$, and* conversion functions $\varphi_j : \mathbb{F}^{c_j} \to \mathbb{F}^{b_j}$*, $j \in [k]$, where $c_j = \sum_{i=1}^{\ell} |\{v \in [k_0] : j \in \psi_i(v)\}|$, such that*

- ***Correctness:*** *For every $(y_i^{(j)})_{i\in[\ell],j\in[k_0]}$ it holds that*

$$\exists \mathbf{r} \in \mathbb{F}^e : (\varphi_j(\mathbf{Y}(j)))_{j=1}^k = \mathsf{Share}_\mathcal{L}\left(\left(\sum_{v=1}^{k_0} y_1^{(v)}, \ldots, \sum_{v=1}^{k_0} y_\ell^{(v)}\right), \mathbf{r}\right).$$

  where $\mathbf{Y}(j) = (y_i^{(v)})_{i\in[\ell],v\in[k_0]:j\in\psi_i(v)}$ denotes the view of party $j \in [k]$.

- ***Security:*** *For every $T \subseteq [k]$ such that $|T| \le t$ and every $i \in [\ell]$ we have that*

$$\left|\bigcup_{j\in T} \{v \in [k_0] : j \in \psi_i(v)\}\right| \le k_0 - 1.$$

We further say that the black-box transformation $(\mathcal{L}, (\psi_i)_i, (\varphi_j)_j)$ has rate R if $\mathcal{L}$ has information rate R.

Given a black-box transformation and a compatible HSS, we may readily deduce an HSS with potentially better rate. A formal description of our 3-step black-box approach is given in the following lemma:

Lemma 4 (Black-box transformations applied to HSS)**.** *Suppose there is a (*potentially computationally secure*) $(k_0 - 1)$-private k_0-server HSS Π_0 for $\mathcal{F} = \{\mathcal{X}^m \to \mathbb{F}\}$ with additive reconstruction over $\mathbb{F}$. In addition, let $(\mathcal{L} = (\mathsf{Share}_{\mathcal{L}}, \mathsf{Rec}_{\mathcal{L}}), (\psi_i)_{i\in[\ell]}, (\varphi_j)_{j\in[k]})$ be a black-box transformation with parameters $(t, k, k_0, \ell, \mathbb{F})$ and rate R. Then, there is a (computationally secure if Π is computationally secure) t-private k-server HSS Π for $\mathcal{F}^\ell$ with rate R. Furthermore, in Π server $j \in [k]$ has individual upload cost*[10] *(for all ℓm inputs) of $\sum_{i\in[\ell],v\in[k_0]:j\in\psi_i(v)} L_v$, where L_v is the individual upload cost of the v'th server in Π_0.*

Proof. As discussed above when we went over the general approach, the HSS Π is defined as follows:

1. **Sharing:** Every input $\mathbf{x}_i \in \mathcal{X}^m$, $i = 1, \ldots, \ell$, is secret shared according to the source (k_0-1)-private k_0-server HSS Π_0 to obtain $(x_i^{(v)})_{v=1}^{k_0}$. For $i = 1, \ldots, \ell$, the shares $(x_i^{(v)})_{v=1}^{k_0}$ are replicated according to ψ_i: $x_i^{(v)}$ is sent to all servers in the set $\psi_i(v) \subseteq [k]$.

2. **Evaluation:** Each server $j \in [k]$ holding share $x_i^{(v)}$ computes $y_i^{(v)} := \mathsf{Eval}(f_i, v, x_i^{(v)})$, where $f_i \in \mathcal{F}$ is the function to be evaluated at $\mathbf{x}_i$. Then, each server $j \in [k]$ locally computes $z_j = \varphi_j(\mathbf{Y}(j))$ as its output share.

3. **Reconstruction:** The client computes $\mathsf{Rec}_{\mathcal{L}}(z_1, \ldots, z_k)$.

We first argue that $\mathsf{Rec}_{\mathcal{L}}(z_1, \ldots, z_k) = (f_1(\mathbf{x}_1), \ldots, f_\ell(\mathbf{x}_\ell))$. Indeed, by the correctness of Π_0, we have that $y_i^{(1)} + \cdots + y_i^{(k_0)} = f_i(\mathbf{x}_i)$, and thus, by the correctness of the black-box transformation there is a vector $\mathbf{r}$ such that

$$(z_j)_{j=1}^k = (\varphi_j(\mathbf{Y}(j)))_{j=1}^k = \mathsf{Share}_{\mathcal{L}}((f_1(\mathbf{x}_1), \ldots, f_\ell(\mathbf{x}_\ell)), \mathbf{r}).$$

Hence, applying $\mathsf{Rec}_{\mathcal{L}}$ to $(z_j)_{j=1}^k$ should recover $(f_1(\mathbf{x}_1), \ldots, f_\ell(\mathbf{x}_\ell))$ by definition of the LMSSS. Furthermore, because the rate of the black-box transformation is the same as the information rate of $\mathcal{L}$, and $(z_j)_{j=1}^k$ are shares of $\mathcal{L}$, we conclude the rate of this HSS is also R. In addition, we want to argue that the HSS satisfies (computational or information theoretic) t-privacy. By the $(k_0 - 1)$ privacy assumption of Π_0 we only need to show that every set of t servers receives at most $k_0 - 1$ shares of each $\mathbf{x}_i$. The fact each server $j \in [k]$ receives a share $x_i^{(v)}$ if $j \in \psi_i(v)$, together with the security of the black-box transformation, guarantee this. Finally, the claim about individual upload cost follows because server j receives shares $(x_i^{(v)})_{i\in[\ell],v\in[k_0]:j\in\psi_i(v)}$ and $|x_i^{(v)}| = L_v$. □

Remark 6 (Using Lemma 4 with $t_0 < k_0 - 1$)**.** *If $t = k_0 - 1$ then Lemma 4 can also allow to combine a t_0-private (possibly $t_0 < k_0 - 1$) k_0-server HSS Π_0 for $\mathcal{F}$ with a black-box transformation to obtain a new t_0-private k-server HSS Π for $\mathcal{F}^\ell$. This follows because a black-box transformation with $t = k_0 - 1$* necessarily *replicates at most a single share $y_i^{(v)}$ of y_i to every party.*

[10]Here, individual upload cost stands for the total number of bits an individual server $j \in [k]$ receives. That is, if $x_i^{(j)}$ are the shares of all ℓm inputs, then the individual upload cost is $\sum_{i=1}^{\ell m} |x_i^{(j)}|$.

4.1 Black-Box Transformations with Large k_0

A simple case to consider is one where k_0 can be large compared to t, k. More concretely, fixing t, k, we would like to find $k_0 = k_0(k, t)$ large enough that there is a black-box transformation with the best possible rate $R = R(k, t)$. Note that if such a transformation exists for some $k_0(k, t)$, then it also exists for all $k'_0 > k_0(k, t)$, because we may simply "bunch servers together" in Π_0. As it turns out, picking $k_0 = \binom{k}{t}$ is sufficient, since we can replicate the shares in such a manner as to obtain the usual t-private k-server CNF sharing of $y_1, \ldots, y_\ell$. Then, to characterize the best rate achievable, it only remains to apply Corollary 1 and Lemma 1. Formally, we have the following theorem.

Theorem 8. *Let t, k, ℓ be integers, and let $\mathbb{F}$ be a finite field. Suppose that $b > \ell/k$.*

There is a black-box transformation with parameters $(t, k, k_0 = \binom{k}{t}, \ell, \mathbb{F})$, such that the associated LMSSS $\mathcal{L}$ has output shares in $\mathbb{F}^b$ (and hence information rate $\ell/(kb)$) if and only if there is an $\mathbb{F}$-linear code $C \in (\mathbb{F}^b)^k$ with rate at least $\ell/(kb)$ and distance at least $t + 1$.

Further, for the "if" direction, for every $j \in [k]$, $c_j = \ell\binom{k-1}{t}$, where $c_j = \sum_{i=1}^{\ell} |\{v \in [k_0] : j \in \psi_i(v)\}|$ is as in Definition 11.

Proof. Since the black-box transformation implies a t-private k-server ℓ-LMSSS $\mathcal{L}$, the "only if" direction follows by Lemma 1.

For the "if" direction, we need to construct a black-box transformation. To this end, let $\eta : [k_0] \to \binom{[k]}{t}$ be a bijection which assigns a t-sized subset of $[k]$ to each index of k_0. For every $i \in [\ell]$ we let $\psi_i(v) = [k] \setminus \eta(v)$ (this already implies $c_j = \ell\binom{k-1}{t}$). Next, let $\mathcal{L}$ be the t-private k-server ℓ-LMSSS $\mathcal{L}$ constructed from C according to Lemma 1 (which has output shares in $\mathbb{F}^b$). For every $i \in [\ell]$, by definition of ψ_i, each server j holds $\mathbf{Y}(j) = (y_i^{(v)})_{i \in [\ell], v \in [K] : j \notin \eta(v)}$. Hence, the servers jointly hold a t-CNF sharing of $y_i^{(1)} + \ldots + y_i^{(k_0)}$. By Corollary 1, this sharing is convertible to $\mathcal{L}$, and we choose φ_j, $j \in [k]$ to be the functions of this conversion.

Correctness follows by the aforementioned observation that $\mathbf{Y}(j)$ are t-CNF shares of $y_i^{(1)} + \ldots + y_i^{(k_0)}$, combined with the fact that the functions φ_j are taken from Corollary 1 to yield a joint sharing of $(y_i^{(1)} + \ldots + y_i^{(k_0)})_{i \in [\ell]}$ according to $\mathcal{L}$.

For security, note that indeed, by definition of η, for every $T \subseteq [k]$ such that $|T| \le t$ we have that $|\bigcup_{j \in T}\{v \in [K] : j \notin \eta(v)\}| = |\{A \in \binom{[k]}{t} : A \cap T \neq \emptyset\}| = |\{B \in \binom{[k]}{k-t} : B \cup T \neq [k]\}| \le \binom{k}{t} - 1$. □

As a corollary of Theorem 8, we can amplify the rate of a t-private computationally secure HSS scheme for circuits that relies on a circular-secure[11] variant of the Learning With Errors assumption [DHRW16, BGI+18]. See Appendix A for the relevant definitions.

Corollary 4 (High-rate t-private HSS for circuits)**.** *Let t, k be integers. Suppose there exists a computationally t_0-private k_0-server HSS for circuits, for $k_0 = \binom{k}{t}$ and $t_0 = k_0 - 1$, with additive reconstruction over $\mathbb{F}_2$ and individual upload cost L. Then, there exists a computationally t-private k-server HSS for circuits with ℓ-bit outputs, $\ell = (k - t)\lceil \log_2 k \rceil$, with download rate $1 - t/k$ and individual upload cost $\ell\binom{k-1}{t}L$.*

Remark 7. *Note that the conclusion of Corollary 4 circumvents the $1 - dt/k$ barrier in Theorem 7 by allowing nonlinear (and computational) HSS.*

[11] Alternatively, circular security can be eliminated if the share size is allowed to grow with the size of the circuit given as input to Eval.

4.2 Black-Box Transformations with $k_0 = 2$

In this section, we consider black-box transformations with $k_0 = 2$. We show that, for the special case of $t = 1$, we may take $k_0 = 2$ and obtain a black-box transformation with *optimal* rate $1 - 1/k$ (for any k). Note that this is better than the guarantee of Theorem 8, which says that if we choose $k_0 = \binom{k}{1} = k$ then we can achieve optimal rate. The fact that we may obtain optimal rate already with $k_0 = 2$ may be surprising since when $k_0 = 2$, essentially only a 1-private 2-server HSS scheme Π_0 can be combined with the black-box transformation.

Theorem 9 (Optimal black-box transformations with $k_0 = 2$)**.** *Let $k \geq 2$ be an integer and $\mathbb{F}$ be a finite field. There is a black-box transformation with parameters $(1, k, k_0 = 2, \ell = k - 1, \mathbb{F})$ and rate $1 - 1/k$. Furthermore, for every $j \in [k]$, $c_j = k - 1$, where $c_j = \sum_{i=1}^{\ell} |\{v \in [k_0] : j \in \psi_i(v)\}|$ is as in Definition 11.*

Proof. For every $i = 1, \dots, k - 1$ we define

$$\psi_i(v) = \begin{cases} [k] \setminus \{i\}, & v = 1 \\ \{i\}, & v = 2. \end{cases}$$

In addition, note that for every server $j = 1, \dots, k - 1$ we have

$$\mathbf{Y}(j) = ((y_i^{(1)})_{i \neq j}, (y_j^{(2)}))$$

and for server $j = k$ we have

$$\mathbf{Y}(k) = (y_i^{(1)})_{i \in [k-1]}$$

This implies that $c_j = k - 1$ for every $j \in [k]$. In addition, for every server $j = 1, \dots, k - 1$ we choose

$$\varphi_j(\mathbf{Y}(j)) = y_j^{(2)} - \sum_{i \neq j} y_i^{(1)}$$

and for server $j = k$ we choose

$$\varphi_k(\mathbf{Y}(k)) = \sum_{i=1}^{k-1} y_i^{(1)}.$$

To prove the correctness of the black-box transformation, consider the LMSSS $\mathcal{L} = (\mathsf{Share}_{\mathcal{L}}, \mathsf{Rec}_{\mathcal{L}})$ over $\mathbb{F}$ defined by $\mathsf{Share}_{\mathcal{L}}((x_1, \dots, x_{k-1}), r) = (x_1 + r, \dots, x_{k-1} + r, -r)$ and $\mathsf{Rec}_{\mathcal{L}}(z_1, \dots, z_k) = (z_1 - z_k, z_2 - z_k, \dots, z_{k-1} - z_k)$. It is not difficult to see that this is a k-server $(k-1)$-LMSSS. To see that the black-box transformation is correct with respect to $\mathcal{L}$, note that the output shares take the form

$$\left(y_1^{(2)} - \sum_{i \neq 1} y_i^{(1)}, y_2^{(2)} - \sum_{i \neq 2} y_i^{(1)}, \dots, y_{k-1}^{(2)} - \sum_{i \neq k-1} y_i^{(1)}, \sum_{i=1}^{k-1} y_i^{(1)} \right).$$

Letting $r = -\sum_{i=1}^{k-1} y_i^{(1)}$, we see that for any $j \in [k-1]$, the j'th output share is

$$y_j^{(2)} - \sum_{i \neq j} y_i^{(2)} = y_j^{(2)} + y_j^{(1)} + r = x_j + r,$$

as desired. By construction, the information rate of the black box scheme is $1 - 1/k$, as we transmit k symbols to reconstruct $k - 1$. Security follows because every server receives exactly one share of each secret. $\square$

Before proceeding with implications of Theorem 9, we observe that any 1-private k-server LMSSS has rate at most $1 - 1/k$ (see Theorem 7). In this sense, Theorem 9 obtains the optimal rate.

Next, we discuss the implications of Theorem 9. Combining Theorem 9 with Proposition 4, we obtain several corollaries. In particular, this can be immediately applied to the constructions in [BGI15, BGI16b], to obtain computationally secure PIR with rate approaching 1 and upload cost $O(\lambda \log N)$, where λ is a security parameter (concretely, the seed length of a length-doubling pseudorandom generator), assuming one way functions exist.

Corollary 5 (High-rate CPIR with logarithmic upload cost). *Suppose one-way functions exist. Then, for any $k \geq 2$, $w \geq 1$, and $\ell = w(k-1)$, there is a computationally 1-private k-server PIR protocol for databases with N records of length ℓ, with upload cost $O(k\lambda \log N)$ and download rate $1 - 1/k$.*

The above can be extended to high-rate HSS schemes for other simple classes of functions, considered in [BGI15, BGI16b], based on one-way functions. Similarly, we can get high-rate variants of 2-server HSS schemes for *branching programs* based on DDH [BGI16a][12] or DCR [OSY21, RS21]. Analogously to Corollary 5, we can get similar HSS schemes for branching programs with rate $1 - 1/k$.

4.3 High-Rate Information-Theoretic PIR with Sub-Polynomial Upload Cost

While Theorem 6 provides PIR protocols with download rate approaching 1 as $k \to \infty$, these schemes have polynomial ($O(N^{1/d})$) upload cost in the database size N. In this section we will construct PIR protocols with download rate approaching 1 as $k \to \infty$ and that have sub-polynomial ($N^{o(1)}$) upload cost, albeit at a worse trade-off between the number of servers and the download rate.

The black-box approaches of Sections 4.1 and 4.2 are not sufficient because they either require an HSS with a high ratio of security-to-servers ($1-1/k_0$), or a very small number of servers ($k_0 = 2$), while in this section we eventually apply it to a 1-private 6-server HSS. We don't obtain a tight result in this section, but, nevertheless, it allows us to convert a t_0-private k_0-server HSS Π_0 to a t_0-private k-server HSS Π with higher rate, where $k > k_0$, and hence it is sufficient for our purposes of improving the rate of PIR protocols with sub-polynomial upload cost.

Note that we preserve the t_0 privacy of the original HSS (in similar fashion to the $k_0 = 2$ case where $t_0 = 1$). Thus, in view of Remark 6, without loss of generality we can assume that $t_0 = k_0 - 1$. To demonstrate the idea of this black-box transformation, suppose we are given a 3-additive secret sharing for every secret $y_i = y_i^{(1)} + y_i^{(2)} + y_i^{(3)}$, $i \in [2]$. We will show how to obtain a 2-private 5-server 2-LMSSS sharing with information rate $2/5 > 1/3$. Let $q = 3$ and suppose we have servers indexed by the elements of $[q] = \{1, 2, 3\}$. In addition, suppose we have two additional servers indexed by the *sets* $\{1, 2\}$ and $\{1, 3\}$, respectively. In this construction we give the third share of the first secret $y_1^{(3)}$ to server $\{1, 2\}$ and distribute the other shares $y_1^{(1)}, y_1^{(2)}$ arbitrary among the servers indexed with $1, 2$. Next, suppose we do the same with the servers $1, 3, \{1, 3\}$ and the shares $y_2^{(1)}, y_2^{(2)}, y_2^{(3)}$. Assuming each server simply adds all the shares it gets, we

[12]Known DDH-based constructions of additive HSS for branching programs have an inverse-polynomial failure probability. When amortizing over ℓ instances, one can use an erasure code to make the failure probability negligible in ℓ while maintaining the same asymptotic download rate.

would like that $\varphi_1(\mathbf{Y}(1)) + \varphi_2(\mathbf{Y}(2)) + \varphi_{\{1,2\}}(\mathbf{Y}(\{1,2\})) = y_1$ (and the same with y_2 and servers $1, 3, \{1, 3\}$). Unfortunately, as it currently stands this does not hold because the servers $1, 2, 3$ hold terms related to y_1 and y_2 that do not cancel. To remedy this, suppose we also give server $\{1, 2\}$ all the shares of y_2 servers $1, 2$ received, so it may subtract them from the sum (and the same with server $\{1, 3\}$ and the shares of y_1). Formally:

$$\begin{aligned} \mathbf{Y}(1) &= \{y_1^{(1)}, y_2^{(1)}\} & \varphi_1(\mathbf{Y}(1)) &= y_1^{(1)} + y_2^{(1)} \\ \mathbf{Y}(2) &= \{y_1^{(2)}\} & \varphi_2(\mathbf{Y}(2)) &= y_1^{(2)} \\ \mathbf{Y}(3) &= \{y_2^{(2)}\} & \varphi_3(\mathbf{Y}(3)) &= y_2^{(2)} \\ \mathbf{Y}(\{1,2\}) &= \{y_1^{(3)}, y_2^{(1)}\} & \varphi_{\{1,2\}}(\mathbf{Y}(\{1,2\})) &= y_1^{(3)} - y_2^{(1)} \\ \mathbf{Y}(\{1,3\}) &= \{y_2^{(3)}, y_1^{(1)}\} & \varphi_{\{1,3\}}(\mathbf{Y}(\{1,3\})) &= y_2^{(3)} - y_1^{(1)} \end{aligned}$$

This satisfies $\varphi_1(\mathbf{Y}(1)) + \varphi_2(\mathbf{Y}(2)) + \varphi_{\{1,2\}}(\mathbf{Y}(\{1,2\})) = y_1$ (and the same with y_2 and servers $1, 3, \{1, 3\}$). Furthermore, since $|\{1, 2\} \cap \{1, 3\}| \le 1$, this scheme is 2-private, as each server receives at most a single share from each secret. Generalizing this idea leads to the following lemma:

Lemma 5. *Let $k_0 < q$ be an integer and $\mathbb{F}$ a finite field. Suppose we have a collection of (k_0-1)-sized subsets from $[q]$, $\mathcal{S} \subseteq \binom{[q]}{k_0-1}$, with the additional requirement that any two distinct sets $S_1, S_2 \in \mathcal{S}$ satisfy $|S_1 \cap S_2| \le 1$. Then there is a black-box transformation with parameters $(k_0 - 1, q + |\mathcal{S}|, k_0, |\mathcal{S}|, \mathbb{F})$ and rate $1 - q/(q + |\mathcal{S}|)$.*

Furthermore, for every $j \in [q + |\mathcal{S}|]$, $c_j \le |\mathcal{S}|$, where e $c_j = \sum_{i=1}^{\ell} |\{v \in [k_0] : j \in \psi_i(v)\}|$ is as in Definition 11.

Proof. Associate indices $i = 1, \dots, \ell$, $\ell = |\mathcal{S}|$, with the sets $S_i \in \mathcal{S}$, and the indices $\ell + 1, \dots, \ell + q$ with the elements of $[q]$. We will describe the construction of $(\psi_i)_{i\in[\ell]}$ in algorithmically as follows.

1. For every $i \in [q + \ell]$ and $v \in [k_0]$ initialize $\psi_i(v) = \emptyset$.
2. For every $i \in [\ell]$, update $\psi_i(k_0) = \psi_i(k_0) \cup \{i\}$.
3. For every $i \in [\ell]$, let $S_i = \{j_1, \dots, j_{k_0-1}\} \in \mathcal{S}$ be the associated set. Update $\psi_i(r) = \psi_i(r) \cup \{\ell + j_r\}$ for $r = 1, \dots, k_0 - 1$ (note that the indices in S_i may be ordered arbitrarily).
4. For every $i \in [\ell]$ let again $S_i = \{j_1, \dots, j_{k_0-1}\} \in \mathcal{S}$ be the associated set. Now, for $i' \in [\ell] \setminus \{i\}$ and $r = 1, \dots, k_0 - 1$ update $\psi_{i'}(r) = \psi_{i'}(r) \cup \{i\}$.

Next, we define $(\varphi_j)_{j\in[q+\ell]}$ as follows:

- For every server $j = 1, \dots, \ell$ let $I_j \subseteq [\ell] \times [k_0]$ denote the set of indices $(i, v) \in I_j$ such that $j \in \psi_i(v)$, *excluding* (j, k_0). Then, we set $\varphi_j(\mathbf{Y}(j)) = y_j^{(k_0)} - \sum_{(i,v)\in I_j} y_i^{(v)}$.
- Every server $j = \ell + 1, \dots, \ell + q$ simply computes the sum of the shares it got $\varphi_j(\mathbf{Y}(j)) = \sum_{i\in[\ell],v\in[k_0]:j\in\psi_i(v)} y_i^{(v)}$.

To show that the above describes a black-box transformation over $\mathbb{F}$, we need to establish security and correctness. We first argue security, which holds because, for every $i \in [\ell]$, every server with index $j \in [q + \ell]$ receives at most a single element $y_i^{(v)}$ (which also implies $c_j \le \ell$). To see this, note that $y_i^{(k_0)}$ is sent exclusively to server with index i (and no other share $y_i^{(v)}$ is sent to it). In addition, every server with index $j = \ell+1, \ldots, \ell+q$ receives at most a single element $y_i^{(v)}$ for every $i \in [\ell]$. It is potentially only possible for a server with index $j \in [\ell]$ to receive elements $y_i^{(v)}, y_i^{(v')}$ with $v \neq v'$. However, if $j = i$ this does not happen because we exclude it in step 4. If $j \neq i$ it can only happen if there are two elements $j_v, j_{v'} \in S_i \cap S_{i'}$ for some $i' \neq i$. However, because we require that $|S_i \cap S_{i'}| \leq 1$ this does not happen. Hence, every subset T of size $|T| \leq k_0 - 1$ satisfies that there is no $i \in [\ell]$ for which the servers in T hold all the shares $y_i^{(1)}, \ldots, y_i^{(k_0)}$.

To argue correctness we first define a $(q+\ell)$-server ℓ-LMSSS over $\mathbb{F}$, $\mathcal{L} = (\mathsf{Share}_{\mathcal{L}}, \mathsf{Rec}_{\mathcal{L}})$ as follows:

- $\mathsf{Share}_{\mathcal{L}}$ first additively shares $y_i = y_i^{(1)} + \ldots + y_i^{(k_0)}$ for every $i \in [\ell]$. Then, each server j receives share $y_i^{(v)}$ if $j \in \psi_i(v)$.
- For $\mathsf{Rec}_{\mathcal{L}}$ note that $\varphi_i(Y(i)) + \sum_{j \in S_i} \varphi_{\ell+j}(Y(\ell+j)) = y_i^{(1)} + \ldots + y_i^{(k_0)} = y_i$ because if $y_{i'}^{(v')} \in Y(\ell+j)$ for $i' \neq i$ then also $y_{i'}^{(v')} \in Y(i)$ (and so these terms cancel as they appear with a minus sign in $\varphi_i(Y(i))$). Therefore, only terms from $\{y_i^{(v)}\}_{v \in [k_0]}$ don't cancel. Because we can reconstruct every y_i, this fully specifies $\mathsf{Rec}_{\mathcal{L}}$.

Now, since $\mathcal{L}$ is an LMSSS and the shares $(\varphi_j(Y(j)))_{j \in [q+\ell]}$ are shares of $\mathcal{L}$, correctness follows. Finally, for the rate of the black-box transformation, note that the information rate of $\mathcal{L}$ is $1 - q/(q+\ell)$. □

Next, we need the PIR protocol from [BIKO12] (although the PIR from [Efr09], which has larger upload cost, can also be used).

Theorem 10. *[BIKO12] There exists a 1-private 3-server HSS* $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ *for* $\mathsf{ALL}_{\mathbb{F}_4}$ *with upload cost* $O\left(2^{6\sqrt{\log N \log\log N}}\right)$. *Furthermore, the output shares satisfy* $z^{(1)} + z^{(2)} + z^{(3)} = \eta \cdot f(x)$, *where* $f \in \mathsf{ALL}_{\mathbb{F}_4}$ *is the function on which* Eval *is applied,* x *is the input, and* $\eta \neq 0$ *depends only on* x *and the randomness of* Share.

The PIR from Theorem 10 does not have additive reconstruction, but its reconstruction algorithm is simple enough so that it can be modified into an additive PIR, at the cost of doubling the number of servers.

Proposition 2. *There exists a 1-private 6-server HSS* Π *for* $\mathsf{ALL}_{\mathbb{F}_4}$ *with additive reconstruction and upload cost* $O\left(2^{6\sqrt{\log N \log\log N}}\right)$.

Proof. Let $\Pi_0 = (\mathsf{Share}_0, \mathsf{Eval}_0, \mathsf{Rec}_0)$ be the PIR from Theorem 10. The HSS Π proceeds as follows:

- **Sharing:** Let $(x_0^{(1)}, x_0^{(2)}, x_0^{(3)}) = \mathsf{Share}_0(x)$ and $\eta \in \mathbb{F}_4$ be as in Theorem 10. Let $\eta^{-1} = \eta_1 + \eta_2$ be an additive sharing. We use 6 servers with input shares: $(x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, x^{(5)}, x^{(6)}) = ((x_0^{(1)}, \eta_1), (x_0^{(2)}, \eta_1), (x_0^{(3)}, \eta_1), (x_0^{(1)}, \eta_2), (x_0^{(2)}, \eta_2), (x_0^{(3)}, \eta_2))$.
- **Evaluation:** Each server r holding a share $x_0^{(i)}$ and η_j, outputs $z_r = \eta_j \cdot \mathsf{Eval}(f, i, x_0^{(i)})$.
- **Reconstruction:** The reconstruction algorithm computes $z_1 + \ldots + z_6$.

Privacy holds because each server receives a single share $x_0^{(i)}$ and a single share η_j. Correctness holds because $\sum_{r=1}^{6} z_r = (\eta_1 + \eta_2) \sum_{i=1}^{3} \mathsf{Eval}(f, i, x_0^{(i)}) = \eta^{-1}\eta f(x) = f(x)$. □

Theorem 11. *There exists a* 1*-private* k*-server PIR for* $2w \cdot (k - \Theta(\sqrt{k}))$*-bit,* $w \in \mathbb{N}$*, record databases of size* N*, with upload cost* $O\left(k^2 \cdot 2^{6\sqrt{\log N \log\log N}}\right)$ *and download cost* $2w \cdot k$*. Consequently, the rate of the PIR is* $1 - 1/\Theta(\sqrt{k})$*.*

Proof. We will first need the following fact.

Fact 4 ([EH63])**.** *Let* $k_0 > 0$ *be an integer. There exists a function* $\delta_{k_0} \colon \mathbb{N} \to [0, \infty)$ *such that* $\lim_{q\to\infty} \delta_{k_0}(q) = 0$ *and such that for every integer* $q > 0$ *there is a family of* $(k_0 - 1)$*-sized subsets from* $[q]$*,* $\mathcal{S}_q \subseteq \binom{[q]}{k_0-1}$*, with the additional requirement that any two distinct sets* $S_1, S_2 \in \mathcal{S}_q$ *satisfy* $|S_1 \cap S_2| \leq 1$*, and such that* $|\mathcal{S}_q| = (1 - \delta_{k_0}(q))\frac{q(q-1)}{k_0(k_0-1)}$*.*

Now, we combine the above fact, Lemma 5, Proposition 2, and the modification to Lemma 4 given in Remark 6, which yields our claim with $w = 1$. To obtain it for $w > 1$ we can just repeat the scheme w times. □

Moreover, similar results for sub-polynomial upload cost PIR protocols with $(t \geq 2)$-privacy could also be obtained. These protocols exist, for example, by combining the results of [BIW07] with the protocols from [Yek07, Efr09, BIKO12].

5 Improving Rate via Nonlinear Reconstruction and a Small Failure Probability

In Section 3.2, we saw that that the rate of $1 - dt/k$ is the best possible for linear HSS. In this section, we will see that if we allow non-linear reconstruction with an exponentially small failure probability, then we can in fact beat this bound.

Remark 8 (Efficiency of Rec)**.** *For this section, we do not attempt to address the computational efficiency of the* Rec *algorithm in our main theorem statement. We note however that there are efficient algorithms for a simpler scheme that still can beat the* $1 - dt/k$ *barrier; see Remark 12.*

Our main theorem, Theorem 12, offers stronger guarantees on the download rate than the simple scheme in Remark 12. The tools that we use to prove it are information-theoretic—the approach is based on classical Slepian-Wolf coding—and do not immediately yield efficient algorithms for Rec*. However, we are hopeful that progress in efficient implementations of Slepian-Wolf coding, e.g. [GFC06, CHJ09, SMT15], may be applicable in our setting as well.*

A warm-up. To motivate how this relaxation to non-linear schemes might help, consider the following simple example. Suppose that $m = d$ and we would like to construct a 1-private HSS for $\{f\}^\ell$, where $f(x_1, \ldots, x_d) = \prod_{i=1}^{d} x_i$, based on Shamir sharing.

Let $k = d + 1$. Let q be a prime power so that $q > k$ and suppose that $|\mathbb{F}| = q \approx k$. Given inputs $x_{i,r} \in \mathbb{F}$ for $i \in [d]$ and $r \in [\ell]$, each server j holds the input shares $(p_{i,r}(\alpha_j))_{i,r}$, where $p_{i,r}$

is a random degree 1 polynomial over $\mathbb{F}$, with $p_{i,r}(0) = x_{i,r}$. Then each party computes the output shares

$$\mathsf{Eval}(f, j, (p_{i,r}(\alpha_j))_{i,r}) = \left(\prod_{i=1}^{d} p_{i,r}(\alpha_j)\right)_{r\in[\ell]} =: \mathbf{y}^j \in \mathbb{F}^\ell.$$

Then, since $k > d$, we may recover $\prod_{i\in[d]} x_{i,r}$ for each $r \in [\ell]$ from the output shares by polynomial interpolation. The download cost of this scheme is $\ell k \log_2(q)$.

Now consider the following improvement on the same scheme. We observe that for $r \in [\ell]$,

$$\Pr[y_r^j = 0] = (1 - 1/q)^d \approx 1/e.$$

Thus, with high probability, there are only about ℓ/e nonzero entries in each $\mathbf{y}^j$. This allows each party to compress their output shares. Rather than sending $\mathbf{y}^j$ itself, each party can send the location and value of each of the nonzeros of $\mathbf{y}^j$. With high probability, the number of bits that each party needs to send is

$$\log_2\binom{\ell}{\ell/e} + \frac{\ell\log_2(q)}{e} = \ell\left((1 + o(1))H(1/e) + \frac{\log_2(q)}{e}\right).$$

For large d (with $k = d + 1$ and $q \approx k$), this is a savings of about a factor of e over the naive version.

A general methodology for HSS share compression. The previous example already shows that we can surpass the barrier of $1 - dt/k$ for linear schemes from Theorem 7 by using a simple compression technique. However, it only applies to evaluating products over a big finite field, which (despite being natural) is not useful for any applications we are aware of. Moreover, this naive compression method is entirely *local*, not taking advantage of correlations between output shares. In the following we develop a more general framework for compressing HSS shares by using Slepian-Wolf coding. We apply this methodology to a simple and well-motivated instance of HSS, where f computes $\mathbb{F}_2$ multiplication (i.e., the AND of two input bits). This can be motivated a "dense" variant of the private set intersection problem, where the sets are represented by their characteristic vectors. We further show that, unlike in the above warm-up example, here it is possible to obtain improved rate while ensuring that the output shares reveal no additional information except the output.

The rest of this section is organized as follows. In Section 5.1, we show that it is possible to push the above line of reasoning as far as the classical *Slepian-Wolf theorem* allows. As described in Section 1.2.3, the Slepian-Wolf theorem describes the extent to which we can compress (possibly correlated) sources that are held by different parties. We cannot use the Slepian-Wolf theorem directly because the underlying joint distribution is not known to each server (as it depends on the value of the secret), but we show how to adapt the proof to deal with this. We apply this general methodology to optimize the download rate of 3-server HSS for AND ($\mathbb{F}_2$ multiplication).

Next, in Section 5.2, we consider a desirable *symmetric privacy* property, ensuring that the output shares reveal no information about the input beyond the output. While symmetric privacy is easy to achieve for HSS with linear reconstruction, this is not necessarily true for HSS with nonlinear reconstruction. Indeed, we show that Shamir-based HSS in the above warm-up example does not satisfy this property. However, somewhat unexpectedly, we show that the optimized HSS for AND from Section 5.1 does satisfy this requirement.

5.1 Beating the $1 - dt/k$ Barrier via Slepian-Wolf Coding

In this section we introduce a broad class of non-linear, $(1 - \exp(-\Omega(\ell)))$-correct HSS schemes for function classes of the form $\mathcal{F}^\ell$ when ℓ is large, which can circumvent the linear impossibility result of Theorem 7 for linear, 1-correct HSS schemes.

5.1.1 Notation and theorem statements

We begin by setting up some further notation for this section. Suppose we have a function class $\mathcal{F}$, and suppose we have a k-server HSS scheme $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ for $\mathcal{F}$. We will formally state how we use Π below when we define our new HSS scheme Π'; for now we sketch enough of it to set up our notation. Given $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \mathcal{X}^m$, we run the Share and Eval functions of Π independently to obtain ℓ output shares (outputs of Eval) for each server j. Let $z_{j,i} \in \mathcal{Z}_j$ be the i'th output share held by server j, for $j \in [k]$ and for $i \in [\ell]$. Let $\mathbf{Z}$ be the $k \times \ell$ matrix whose (j, i) entry is $z_{j,i}$. We refer to the columns of $\mathbf{Z}$ as $\mathbf{z}_i$ and the rows as $\mathbf{y}_j$. Letting

$$\mathcal{Z} = \mathcal{Z}_1 \times \mathcal{Z}_2 \times \cdots \times \mathcal{Z}_k,$$

we also view $\mathbf{Z}$ as an element of $\mathcal{Z}^\ell$, that is, as a sequence of the columns of the matrix $\mathbf{Z}$. Abusing notation, we will move back and forth between these two views.

Each $\mathbf{x} \in \mathcal{X}^m$ induces a distribution on $\mathbf{z} \in \mathcal{Z}$, obtained by first using Share to share $\mathbf{x}$ among the k servers, and then using Eval to arrive at output shares $\mathbf{z}$. The randomness of this distribution comes from the randomness in the probabilistic algorithms Share and Eval. Let $\mathcal{D}$ be the set of all probability distributions on $\mathcal{Z}$ that arise this way. Notice that $|\mathcal{D}| \leq |\mathcal{X}|^m$, the number of possible values for $\mathbf{x}$.

We define $\Delta(\mathcal{D})$ to be the collection of probability distributions on $\mathcal{D}$. We define $\Delta_\ell(\mathcal{D}) \subset \Delta(\mathcal{D})$ to be the collection of distributions with denominator ℓ. That is,

$$\Delta_\ell(\mathcal{D}) = \{\pi \in \Delta(\mathcal{D}) : \forall D \in \mathcal{D}, \exists i \in \mathbb{Z}, \pi(D) = i/\ell\}.$$

Notice that

$$|\Delta_\ell(\mathcal{D})| \leq \ell^{|\mathcal{D}|} \leq \ell^{|\mathcal{X}|^m}. \tag{8}$$

For $\pi \in \Delta(\mathcal{D})$, define a distribution σ_π on $\mathcal{Z}$ by

$$\sigma_\pi(\mathbf{z}) := \sum_{D \in \mathcal{D}} \pi(D) \cdot D(\mathbf{z}),$$

where for a distribution D, $D(\mathbf{z})$ denotes the probability of $\mathbf{z}$ under D. Thus, σ_π is defined by first drawing $D \sim \pi$ and then drawing $\mathbf{z} \sim D$.

With all of this notation out of the way, we may state the main result in this section, which states that given a particular HSS scheme $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ for a function class $\mathcal{F}$, we may "compress" the outputs of Eval to create a new HSS scheme for $\mathcal{F}^\ell$, for sufficiently large ℓ, with potentially better download rate. As noted in Section 1.2.3, this theorem is closely related to the classical Slepian-Wolf theorem for compression of dependent sources with separate encoders and a joint decoder.

Theorem 12 (Slepian-Wolf coding for HSS shares)**.** *Suppose there is a t-private k-server HSS $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ for $\mathcal{F}$. Let $\mathcal{D}$ be defined as above. There is a function $\delta : \mathbb{R} \to \mathbb{R}$ so that*

$\delta(\varepsilon) \to 0$ as $\varepsilon \to 0$, and constants C and ℓ_0 (all of which may depend on Π and its parameters) so that the following holds for any $\varepsilon > 0$ and any $\ell \geq \ell_0$.

For a distribution $\pi' \in \Delta(\mathcal{D})$, let $\mathbf{z}' \in \mathcal{Z}$ denote a random variable drawn from $\sigma_{\pi'}$. Suppose that $b_1, b_2, \ldots, b_k$ are non-negative integers so that for all $S \subseteq [k]$,

$$\sum_{i \in S} b_i \geq \max_{\pi' \in \Delta(\mathcal{D})} \ell \cdot (H(\mathbf{z}'_S | \mathbf{z}'_{S^c}) + \delta(\varepsilon)).$$

Then there is a t-private k-server α-correct HSS scheme for $\mathcal{F}^\ell$ with $\alpha = 1 - \exp(-C\varepsilon^2 \ell)$, and with download cost $\sum_{i=1}^k b_i$.

Remark 9. *In order to interpret the bound on the b_i's in Theorem 12, let us consider it in terms of the download cost of the naive scheme that repeats Π ℓ times. Suppose that, in the original scheme Π, server i sends b_i bits, so server i sends ℓb_i bits in the naive scheme. Since $\mathbf{z}'_S \in \{0,1\}^{\sum_{i \in S} b_i}$, we (naively) have*

$$H(\mathbf{z}'_S | \mathbf{z}'_{S^c}) \leq \sum_{i \in S} b_i$$

for all $S \subseteq [k]$, and the total download cost is $\sum_{i=1}^k b_i$. Thus (ignoring the slack $\delta(\varepsilon)$), the naive bound on $H(\mathbf{z}'_S | \mathbf{z}'_{S^c})$ yields the download cost of the naive scheme. As we will see below, by obtaining better-than-naive bounds on this conditional entropy term, we will be able to beat the naive scheme, resulting in a new scheme that has an improved download rate.

Remark 10. *Theorem 12 is tight in the sense that if we are to follow the strategy of "compress the output shares of a fixed HSS Π," $\max_{\pi'} \ell \cdot H(\mathbf{z}'_S | \mathbf{z}'_{S^c})$ is the best download cost we can hope for. This follows from the standard converse for Slepian-Wolf coding, by allowing an adversary to choose inputs that yield a sequence $D_1, \ldots, D_\ell$ of distributions in $\mathcal{D}$ with empirical distribution that is approximately the maximizer π^*.*

Before we prove the theorem, we apply it to a simple case where we would like to multiply two bits. We will apply it to the following HSS.

Definition 12 (Greedy-Monomial CNF HSS)**.** *Let t, k, d, m be positive integers with $k > dt$ and let $\mathbb{F}$ be a finite field. Define a t-private k-server HSS $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ for $\mathsf{POLY}_{d,m}(\mathbb{F})$ as follows.*

- ***Sharing.*** *The* Share *function is given by t-CNF sharing. To set notation, suppose that server j receives $\mathbf{y}^j = (X_{i,S} : j \notin S)$ where $X_{i,S}$ for $i \in [m]$ and $S \subset [k]$ of size t are random so that $\sum_S X_{i,S} = x_i$.*
- ***Evaluation.*** *Let $f \in \mathsf{POLY}_{d,m}(\mathbb{F})$. We may view $f(x_1, \ldots, x_m)$ as a polynomial $F(\mathbf{X})$ in the variables $\mathbf{X} = (X_{i,S})_{i \in [m], S \subset [k]}$. Each server j can form some subset of the monomials $\prod_{s=1}^r X_{i_s, S_s}$ that appear in $F(\mathbf{X})$. Server 1 greedily assembles all of the monomials in $F(\mathbf{X})$ that they can; the sum of these monomials is $\mathsf{Eval}(f, 1, \mathbf{y}^1)$. Inductively, Server j greedily assembles all of the monomials in $F(\mathbf{X})$ that they can and that have not been taken by Servers $1, \ldots, j-1$, and the sum of these monomials is $\mathsf{Eval}(f, j, \mathbf{y}^j)$.*
- ***Reconstruction.*** *By construction, $f(x_1, \ldots, x_m)$ is equal to $\sum_j \mathsf{Eval}(f, j, \mathbf{y}^j)$. Thus,* Rec *is defined additively.*

We refer to this Π *as the t-private, k-server* greedy monomial CNF *HSS.*

Corollary 6. *Let $\mathcal{F} = \{f(x_1, x_2) \mapsto x_1 \cdot x_2\}$. Let $m = 2$ and $\mathcal{X} = \mathcal{Y} = \mathbb{F}_2$, so $f : \mathcal{X}^m \to \mathcal{Y}$. For sufficiently large ℓ, the Greedy Monomial CNF HSS is a 3-server, 1-private, $\left(1 - 2^{-\Omega(\ell)}\right)$-correct HSS for the function family $\mathcal{F}^\ell$, with download rate $R \geq 0.376$.*

Moreover, for these parameters, the Greedy Monomial CNF HSS yields the best download rate when plugged into Theorem 12, out of all $\mathbb{F}_2$-linear HSS schemes.

Remark 11 (Greedy Monomial CNF admits optimal compression for 3-server AND)**.** *Corollary 6 shows that the simple Greedy Monomial CNF HSS can be modified to exceed the $1 - dt/k$ limit of Theorem 7 for linear schemes, even in the extremely bare-bones case of multiplying two bits. (Notice that the Shamir-based example at the beginning of Section 5 requires large fields). Moreover, the "moreover" part of the corollary, along with Remark 10, suggests that the the rate 0.376 is the best achievable by these methods for this problem. In comparison, a symmetric monomial assignement that assigns 3 monomials to each server achieves rate ≈ 0.350.*

Remark 12 (A simpler way to get the same qualitative result)**.** *In fact, we beat the $1 - dt/k$ bound without the full power of Theorem 12, even over $\mathbb{F}_2$. Rather than compressing the joint distribution of the output shares, each server may compress their share individually. Notice that by the privacy guarantee, the distribution of each server's share does not depend on the secret; thus, if this distribution is not uniform, it can be compressed. As ℓ grows, the download rate achievable can approach*

$$\frac{\log_2 |\mathcal{Y}|}{\sum_{j=1}^{k} H(y^j)}, \tag{9}$$

where y^j is the output share for party j.

This straightforward scheme, while it does not yield bounds as strong as those obtained by Theorem 12, already allows us to beat the $1 - dt/k$ bound of Theorem 7. For example, for the 3-party Greedy Monomial CNF scheme for multiplying two bits, the rate in (9) is easily seen to be $R \approx 0.367 > 1/3$.

Further, this scheme immediately comes with efficient Eval *and* Rec *algorithms, using known efficient algorithms for optimal compression (for example Lempel-Ziv coding).*

Proof of Corollary 6. To prove the first part of the corollary, we apply Theorem 12, which states that there is an HSS scheme Π' for $\mathcal{F}^\ell$ of download cost $\sum_i b_i$ provided that, for each $S \subseteq [k]$,

$$\max_{\pi' \in \Delta(\mathcal{D})} H(\mathbf{z}'_S | \mathbf{z}'_{S^c}) \leq \sum_{i \in S} b_i, \tag{10}$$

where $\mathbf{z}' \sim \sigma_\pi$. To prove the theorem we describe a particular HSS scheme to use as Π. Our scheme Π is the 1-private 3-party greedy CNF HSS, as in Definition 12.

We will compute the quantity on the left-hand-side of (10) for Π. We note that when (X, Y) are random variables with joint distribution μ, the function $H(X|Y)$ is concave in μ. This implies that if we view $H(\mathbf{z}'_S | \mathbf{z}'_{S^c})$ as a function of the distribution $\sigma_{\pi'}$ on $\mathbf{z} \in \mathcal{Z}$ that is induced by π', the function is concave in $\sigma_{\pi'}$. Thus, we may solve a convex optimization problem to find the optimal $\sigma_{\pi'}$, and hence the optimal value of (10), for each $S \subseteq [k]$.[13] Implementing this, we obtain the

[13]This approach works for any original HSS scheme Π, but for the particular Π defined above, the situation is even simpler because $\mathcal{D}$ consists of only two distributions. Thus, the problem becomes a univariate convex optimization problem that can be easily solved using calculus.

following requirements on b_0, b_1, b_2.[14]

$$\begin{aligned}
\frac{b_0}{\ell} &\geq 0.75089 \\
\frac{b_1}{\ell} &\geq 0.90690 \\
\frac{b_2}{\ell} &\geq 0.90690 \\
\frac{b_0 + b_1}{\ell} &\geq 1.70429 \\
\frac{b_0 + b_2}{\ell} &\geq 1.70429 \\
\frac{b_1 + b_2}{\ell} &\geq 1.84745 \\
\frac{b_0 + b_1 + b_2}{\ell} &\geq 2.65873
\end{aligned}$$

We see that if we take $b_1 = b_2 = \frac{\ell}{2} \cdot 1.84745 = \ell \cdot 0.92372$ and if we take $b_0 = \ell \cdot 2.65873 - b_1 - b_2 = \ell \cdot 0.81128$ then all of these requirements are satisfied, and we have

$$b_0 + b_1 + b_2 = \ell \cdot 2.65873.$$

Thus, Theorem 12 implies that there is an HSS scheme Π' that approaches this download cost, provided that ℓ is sufficiently large. Thus, for any download rate R satisfying

$$R < \frac{1}{2.65873} \approx 0.37612,$$

there is some large enough ℓ so that there is an HSS scheme for $\mathcal{F}^\ell$ with download rate R. This proves the first part of the corollary.

For the "moreover" part of the corollary, we first observe that any linear HSS scheme is some sort of monomial assignment. Indeed, writing the shares of the inputs x_1, x_2 as $X_{i,j}$ for $i \in [2]$ and $j \in [3]$ (where $x_i = \sum_j X_{i,j}$), then each server's output share is some polynomial in the $X_{i,j}$. Next, we simply enumerate over all such assignments and check the values in Theorem 12.

In more detail, as we assume that the HSS scheme is linear, we must include all monomials of the form $X_{1,j}X_{2,j}$ an odd number of times across the servers. Up to symmetries, there are only two ways to distribute these degree-2 monomials (the way prescribed by the Greedy Monomial CNF assignment, and one other way that assigns three monomials to each server). We must include each linear monomial $X_{i,j}$ an even number of times across the servers, as no degree-1 monomial occurs in $x_1 x_2 = \sum_{i,j} X_{1,i} X_{2,j}$. There are 64 ways to do this (including ways that are equivalent by symmetry). Finally, degree 3 or higher monomials cannot occur in any server, as any such monomial $X_{i_1,j_1} X_{i_2,j_2}, X_{i_3,j_3}$ must contain two distinct values in $\{j_1, j_2, j_3\}$ and thus can only be held by one server and cannot cancel out. Thus, there are only 128 things to check (again, with some redundancy by symmetry). This is easily done with the code referenced earlier. We see that

[14] The code that we used to obtain this can be found at `https://web.stanford.edu/~marykw/files/HSS_SlepianWolf.sage`. This code also obtains methods for finding the requirements on b_i in Theorem 12, starting from more general original HSS schemes Π.

for all of these settings, the condition on $b_1 + b_2 + b_3$ is one of the following three requirements:

$$\frac{b_0 + b_1 + b_2}{\ell} \geq 2.65873$$
$$\frac{b_0 + b_1 + b_2}{\ell} \geq 2.90564$$
$$\frac{b_0 + b_1 + b_2}{\ell} \geq 2.85200$$

The first of these is the binding constraint for the Greedy Monomial CNF scheme. The other two are larger, meaning that there is no linear HSS that can outperform the Greedy Monomial CNF HSS. □

5.1.2 Proof of Theorem 12

As discussed in Section 1.2.3, the statement of Theorem 12 is quite similar to the classical Slepian-Wolf coding theory. The difference between our setup and that of the Slepian-Wolf theorem is that we do not know the underlying distributions, which depend on the secrets. However, our proof follows the basic outline of the classical Slepian-Wolf argument; the main difference is that we need to take a union bound over the distributions $\pi \in \Delta_\ell(\mathcal{D})$ that may arise, and we need to argue that the distribution of output shares arising from a sequence $(D_1, \ldots, D_\ell)$ with empirical distribution π is sufficiently close to i.i.d. samples from σ_π.[15]

We begin with some necessary notions from information theory. We tailor these notions to our notation and setting for readability; in particular, the notation is non-standard from an information-theory point of view.

We use the notation from the beginning of Section 5.1.1. Given $\mathbf{Z} \in \mathcal{Z}^\ell$ and given $\mathbf{z} \in \mathcal{Z}$ we write $f_{\mathbf{Z}}(\mathbf{z})$ to denote the number of times that $\mathbf{z}$ appears in $\mathbf{Z}$. (Viewing $\mathbf{Z}$ as a matrix, this is the number of times that $\mathbf{z}$ appears as a column of $\mathbf{Z}$).

For a set $S \subset [k]$, let $\mathcal{Z}_S = \prod_{j \in S} \mathcal{Z}_j$ (recalling that $\mathcal{Z} = \prod_{j=1}^k \mathcal{Z}_j$). For $\mathbf{z} \in \mathcal{Z}$, we write $\mathbf{z}_S \in \mathcal{Z}_S$ to denote the restriction of $\mathbf{z}$ to the coordinates indexed by S. Given $\mathbf{z}_S \in \mathcal{Z}_S$ and $\mathbf{z}_{S^c} \in \mathcal{Z}_{S^c}$, we denote by $\mathbf{z}^S \circ \mathbf{z}^{S^c}$ the vector $\mathbf{z} \in \mathcal{Z}$ obtained by combining the two of them in the natural way (that is, $z_j = z_j^S$ if $j \in S$ and $z_j^{S^c}$ otherwise). Given $\mathbf{Z}_S \in (\mathcal{Z}_S)^\ell$ and $\mathbf{Z}_{S^c} \in (\mathcal{Z}_{S^c})^\ell$, we define $\mathbf{Z}_S \circ \mathbf{Z}_{S^c} \in \mathcal{Z}^\ell$ similarly.

Definition 13. *Let $\varepsilon > 0$. Let σ be a probability distribution on $\mathcal{Z}$. We say that $\mathbf{Z} \in \mathcal{Z}^\ell$ is a* ε-strongly typical sequence *for σ of length ℓ if for all $\mathbf{z} \in \mathcal{Z}$ so that $\sigma(\mathbf{z}) = 0$, we have $f_{\mathbf{Z}}(\mathbf{z}) = 0$; and for all $\mathbf{z} \in \mathcal{Z}$ so that $\sigma(\mathbf{z}) > 0$, we have*

$$|f_{\mathbf{Z}}(\mathbf{z})/\ell - \sigma(\mathbf{z})| \leq \frac{\varepsilon}{|\mathcal{Z}|}.$$

We define

$$T_{\varepsilon,\sigma}^{(\ell)} = \left\{ \mathbf{Z} \in \mathcal{Z}^\ell \, : \, \mathbf{Z} \text{ is } \varepsilon\text{-strongly typical for } \sigma \right\}.$$

Let $S \subset [k]$ and fix $\mathbf{Z}_{S^c} = (\mathbf{z}_{S^c}^1, \ldots, \mathbf{z}_{S^c}^\ell) \in (\mathcal{Z}_{S^c})^\ell$. We define

$$T_{\varepsilon,\sigma,S}^{(\ell)}(\mathbf{Z}_{S^c}) = \left\{ \mathbf{Z}_S \in (\mathcal{Z}_S)^\ell \, : \, \mathbf{Z}_S \circ \mathbf{Z}_{S^c} \in T_{\varepsilon,\sigma}^{(\ell)} \right\}.$$

[15] The empirical distribution π of $(D_1, \ldots, D_\ell) \in \mathcal{D}^\ell$ is sometimes called the *type* of $(D_1, \ldots, D_\ell)$.

That is, $T^{(\ell)}_{\varepsilon,\sigma}$ is the set of all $\mathbf{Z}$ that have about the right number of copies of each $\mathbf{z} \in \mathcal{Z}$, and $T^{(\ell)}_{\varepsilon,\sigma,S}(\mathbf{Z}_{S^c})$ is the set of all ways $\mathbf{Z}_{S^c}$ to complete $\mathbf{Z}_{S^c}$ in order to have that property.

We will use the following well-known fact (see, for example [CT06, Eq. (10.175)]).

Theorem 13. *Let $0 < \varepsilon' < \varepsilon$. Let $S \subseteq [k]$. Suppose that ℓ is sufficiently large. Let σ be a distribution on $\mathcal{Z}$, and let σ_S be the restriction to $\mathcal{Z}_S$. Let $\mathbf{z}' \sim \sigma$ be a random vector drawn from σ. For all $\mathbf{Z}_{S^c} \in \mathcal{Z}^{\ell}_{S^c}$,*

$$|T^{(\ell)}_{\varepsilon,\sigma,S}(\mathbf{Z}_{S^c})| \leq (\ell+1)^{|\mathcal{Z}|} \cdot 2^{\ell H(\mathbf{z}'_S|\mathbf{z}'_{S^c})(1+\delta(\varepsilon))},$$

for some function δ so that $\delta(\varepsilon) \to 0$ as $\varepsilon \to 0$.

With this machinery in place, we can prove Theorem 12.

Proof of Theorem 12. We use the notation listed from the beginning of Section 5.1.1 as well as earlier in this section. Define

$$T^{(\ell)}_{\varepsilon} := \bigcup_{\pi \in \Delta_\ell(\mathcal{D})} T^{(\ell)}_{\varepsilon,\sigma_\pi}.$$

We now define our new HSS scheme, $\Pi' = (\mathsf{Share}', \mathsf{Eval}', \mathsf{Rec}')$. Let $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ be the HSS scheme assumed in the theorem statement.

- **Sharing.** Suppose the inputs are $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \mathcal{X}^m$, where $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,m})$. We define the new Share' function to just use Share to independently share each input $x_{i,r}$. Organize server j's shares into vectors

  $$\mathbf{x}^j_i = (x^j_{i,1}, x^j_{i,2}, \ldots, x^j_{i,m}) \in \mathcal{X}^m$$

  for each $i \in [\ell]$.

- **Evaluation.** Let $f = (f_1, \ldots, f_\ell) \in \mathcal{F}^\ell$. For each $i \in [\ell]$ and for each server $j \in [k]$, we use the Eval function from the original HSS scheme to obtain

  $$z_{j,i} = \mathsf{Eval}(f_i, j, \mathbf{x}^j_i)$$

  Let $\mathbf{Z} \in \mathcal{Z}^\ell$ be the organization of the $z_{j,i}$ as described in Section 5.1.1. Note that we may also think of $\mathbf{Z}$ as a matrix with $\mathbf{Z}_{j,i} = z_{j,i}$. In this notation, server j holds $\mathbf{y}^j$, the j'th row of $\mathbf{Z}$.

  For each server $j \in [k]$, choose a random map[16] $h_j : \mathcal{Z}^\ell_j \to \{0,1\}^{b_j}$. Define a new evaluation function

  $$\mathsf{Eval}'(f, j, (x^j_1, \ldots, x^j_m)) := h_j(\mathbf{y}^j).$$

  For notational convenience later on, let $h(\mathbf{Z}) = (h_1(\mathbf{y}^1), \ldots, h_k(\mathbf{y}^k))$.

- **Reconstruction.** Given output shares $(v^1, \ldots, v^k) \in \{0,1\}^{b_1+\cdots+b_k}$, we wish to reconstruct $f_i(\mathbf{x}_i)$ for $i \in [\ell]$. We define the new reconstruction map, Rec', as follows. Choose a parameter $\varepsilon > 0$.

[16]The servers can have access to the same random map by assuming they share common randomness. Alternatively, we can use pairwise independent hash functions, whose description is short and can be given as part of the input shares.

- If there is a unique $\tilde{\mathbf{Z}} \in T_\varepsilon^{(\ell)}$ with rows $\tilde{\mathbf{y}}^j$ so that $h_j(\tilde{\mathbf{y}}^j) = v^j$ for all $j \in [k]$, define

$$\mathsf{Rec}'(v^1, \ldots, v^k) = \mathsf{Rec}(\tilde{\mathbf{z}}^1), \mathsf{Rec}(\tilde{\mathbf{z}}^2), \ldots, \mathsf{Rec}(\tilde{\mathbf{z}}^\ell),$$

  where $\tilde{\mathbf{z}}^i$ is the i'th column of $\tilde{\mathbf{Z}}$.
- If there is not a unique such $\tilde{\mathbf{Z}}$, define $\mathsf{Rec}'(v^1, \ldots, v^k) = \bot$.

We first note that Π' is t-private, since Π is. We also note that the download cost of Π' is indeed $b_1 + \cdots + b_k$. It remains to show that this scheme is correct with high probability, provided that the b_i satisfy the requirements in the theorem statement.

Suppose that the secrets $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ are shared and evaluated using Share' and Eval' as above. Server j holds $\mathbf{y}^j$, which is organized into $\mathbf{Z} \in \mathcal{Z}^\ell$, as per the definition of Π' above. The i'th columns $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ of $\mathbf{Z}$ is generated according to a distribution $D_i \in \mathcal{D}$ that depends on $\mathbf{x}_i$. Suppose that $(D_1, D_2, \ldots, D_\ell)$ is this vector of distributions, and let $\pi \in \Delta_\ell(\mathcal{D})$ be the empirical distribution of $(D_1, \ldots, D_\ell)$. That is,

$$\pi(D) := \frac{|\{i \in [\ell] \,:\, D_i = D\}|}{\ell}.$$

Define

$$\mathcal{S}_D = \{i \in [\ell] \,:\, D_i = D\}.$$

We first observe that if the correct matrix $\mathbf{Z}$ is identified in the definition of Rec' as the unique $\tilde{\mathbf{Z}} \in T_\varepsilon^{(\ell)}$, then the decoding algorithm Rec' is correct. Indeed, this follows from the correctness of Rec. Thus, to show that this algorithm is correct, it suffices to show that with high probability (over Share, Eval'), the following two things hold.

(A) $\mathbf{Z}$ lies in $T_\varepsilon^{(\ell)}$.

(B) There is no other $\tilde{\mathbf{Z}} \in T_\varepsilon^{(\ell)}$ so that $h(\tilde{\mathbf{Z}}) = h(\mathbf{Z})$.

We begin with (A).

Claim 2. *Let $\mathbf{Z} \in \mathcal{Z}^\ell$ be as above. Then with probability at least $1 - \exp(-\Omega(\varepsilon^2 \ell))$, $\mathbf{Z}$ is ε-strongly typical for σ_π. That is, $\mathbf{Z} \in T_{\varepsilon,\sigma_\pi}^{(\ell)}$.*

Above, the $\Omega(\cdot)$ hides constants that may depend on the original HSS scheme Π and its parameters $(m, k, |\mathcal{X}|, \mathcal{Z})$.

Proof. We must show that $\mathbf{Z}$ is ε-strongly typical for σ_π, or in other words that for all $\mathbf{z} \in \mathcal{Z}$ with $\sigma_\pi(\mathbf{z}) = 0$, we have $f_{\mathbf{Z}}(\mathbf{z}) = 0$; and for all $\mathbf{z} \in \mathcal{Z}$ with $\sigma_\pi(\mathbf{z}) > 0$, we have

$$\left| \frac{f_{\mathbf{Z}}(\mathbf{z})}{\ell} - \sigma_\pi(\mathbf{z}) \right| \leq \varepsilon / |\mathcal{Z}|.$$

To show the first thing, suppose that $\sigma_\pi(\mathbf{z}) = 0$. Then

$$0 = \sigma_\pi(\mathbf{z}) = \sum_{D \in \mathcal{D}} \pi(D) D(\mathbf{z}),$$

and so for each D, either $\pi(D) = 0$, in which case D_i is never equal to D, or else $D(\mathbf{z}) = 0$, in which case there is no way to have $\mathbf{z}_S^i = \mathbf{z}$ if $D_i = D$. In either case, if $\sigma_\pi(\mathbf{z}) = 0$, then $\mathbf{z}$ will never appear in $\mathbf{Z}$ and $f_{\mathbf{z}}(\mathbf{Z}) = 0$.

Next, fix any $\mathbf{z} \in \mathcal{Z}$ so that $\sigma_\pi(\mathbf{z}) > 0$. The number of $\mathbf{z}$ that appear in $\mathbf{Z}$ is

$$f_{\mathbf{Z}}(\mathbf{z}) = \sum_{i=1}^{\ell} \mathbf{1}[\mathbf{z}^i = \mathbf{z}] = \sum_{D \in \mathcal{D}} \sum_{i \in \mathcal{S}_D} \mathbf{1}[\mathbf{z}^i = \mathbf{z}].$$

Recall that we have $\mathbf{z}^i \sim D_i$ for $i \in [\ell]$, each independently (and the distributions $D_i \in \mathcal{D}$ are fixed by the secrets $\mathbf{x}^i$). The expectation of $f_{\mathbf{Z}}(\mathbf{z})$ is thus

$$\begin{aligned}\mathbb{E}[f_{\mathbf{Z}}(\mathbf{z})] &= \sum_{D \in \mathcal{D}} \ell \pi(D) \Pr_D[\mathbf{z}^i = \mathbf{z}] \\ &= \sum_{D \in \mathcal{D}} \ell \pi(D) D(\mathbf{z}) \\ &= \ell \sigma_\pi(\mathbf{z}).\end{aligned}$$

Thus, Hoeffding's inequality implies that, for all such $\mathbf{z}$,

$$\begin{aligned}\Pr\left[\left|\frac{f_{\mathbf{Z}}(\mathbf{z})}{\ell} - \sigma_\pi(\mathbf{z})\right| > \varepsilon/|\mathcal{Z}|\right] &= \Pr\left[\left|\frac{1}{\ell}\sum_{i=1}^{\ell}\left(\mathbf{1}[\mathbf{z}^i = \mathbf{z}] - \mathbb{E}\mathbf{1}[\mathbf{z}^i = \mathbf{z}]\right)\right| > \varepsilon/|\mathcal{Z}|\right] \\ &\leq 2\exp\left(-2\ell\varepsilon^2/|\mathcal{Z}|^2\right)\end{aligned}$$

Since $|\mathcal{Z}|$ is some constant that depends only on the original HSS scheme Π (and in particular, is independent of ℓ and ε), we conclude that for any $\mathbf{z} \in \mathcal{Z}$,

$$\Pr\left[\left|\frac{f_{\mathbf{Z}}(\mathbf{z})}{\ell} - \sigma_\pi(\mathbf{z})\right| > \varepsilon/|\mathcal{Z}|\right] \leq \exp\left(-\Omega(\ell\varepsilon^2)\right).$$

Taking a union bound over all possible $\mathbf{z} \in \mathcal{Z}$, we see that

$$\Pr[\mathbf{Z} \notin T^{(\ell)}_{\varepsilon,\sigma_\pi}] \leq |\mathcal{Z}| \cdot \exp\left(-\Omega(\ell\varepsilon^2)\right) \leq \exp(-\Omega(\ell\varepsilon^2)),$$

using the fact again that $|\mathcal{Z}|$ is a constant. This proves the claim. $\square$

As a corollary of Claim 2, we see that item (A) holds with probability at least $1 - \exp(-\Omega(\ell\varepsilon^2))$. Indeed, with at least that probability we have

$$\mathbf{Z} \in T^{(\ell)}_{\varepsilon,\sigma_\pi} \subseteq T^{(\ell)}_{\varepsilon},$$

using the definition of $T^{(\ell)}_{\varepsilon}$.

Now we turn to item (B), that there is no other $\tilde{\mathbf{Z}} \in T^{(\ell)}_{\varepsilon}$ so that $h(\mathbf{Z}) = h(\tilde{\mathbf{Z}})$. We establish 2^k bad events that could prevent (B) from occurring, one for each $S \subset [k]$.

For $S \subseteq [k]$, define the event $\mathcal{E}_S$ to be the event that there is some $\tilde{\mathbf{Z}}_S \in \mathcal{Z}_S^{\ell}$ so that

- Each row of $\tilde{\mathbf{Z}}_S$ is different than the corresponding row of $\mathbf{Z}_S$. (That is, if we let $\tilde{\mathbf{y}}_i$ denote the i'th row of $\tilde{\mathbf{Z}}_S$, we have $\tilde{\mathbf{y}}_i \neq \mathbf{y}_i$ for all $i \in S$.)
- $\tilde{\mathbf{Z}}_S \circ \mathbf{Z}_{S^c} \in T^{(\ell)}_{\varepsilon}$
- $h(\tilde{\mathbf{Z}}_S \circ \mathbf{Z}_{S^c}) = h(\mathbf{Z})$

Above, we recall the notation that $\tilde{\mathbf{Z}}_S \circ \mathbf{Z}_{S^c}$ means that we should create a matrix in $\mathcal{Z}^\ell$ by taking the rows indexed by S from $\tilde{\mathbf{Z}}$ while taking the rows indexed by S^c from $\mathbf{Z}$.

Observe that (B) will occur provided that none of the bad events $\mathcal{E}_S$ occur. Indeed, suppose that there is some $\tilde{\mathbf{Z}} \in T_\varepsilon^{(\ell)}$ so that $\tilde{\mathbf{Z}} \neq \mathbf{Z}$ and so that $h(\tilde{\mathbf{Z}}) = h(\mathbf{Z})$. Then $\mathcal{E}_S$ occurs, where $S \subseteq [k]$ is the set of rows on which $\mathbf{Z}$ and $\tilde{\mathbf{Z}}$ differ.

Claim 3. *Fix $\mathbf{Z} \in \mathcal{Z}^\ell$ and suppose that the favorable outcome of Claim 2 holds for $\mathbf{Z}$. Let $S \subseteq [k]$. There is some function $\delta'(\varepsilon)$ so that $\delta'(\varepsilon) \to 0$ and $\varepsilon \to 0$, (and which may depend on the HSS Π and its parameters but which is independent of ℓ, ε), so that the following holds.*

For a distribution $\pi' \in \Delta(\mathcal{D})$, let $\mathbf{z}' \in \mathcal{Z}$ denote a random variable drawn from σ'_π. Suppose that

$$\sum_{i \in S} b_i \geq \max_{\pi' \in \Delta(\mathcal{D})} \ell \cdot (H(\mathbf{z}'_S | \mathbf{z}'_{S^c}) + \delta'(\varepsilon)),$$

where the b_i are as in the definition of Share'. Then

$$\Pr[\mathcal{E}_S] \leq \exp(-\Omega(\varepsilon \ell)),$$

where above the $\Omega(\cdot)$ notation suppresses constants that depend on the HSS Π and its parameters.

Proof. Note that for any $\tilde{\mathbf{Z}}_S$ satisfying the first condition of $\mathcal{E}_S$, we have

$$\Pr_h[h(\tilde{\mathbf{Z}}_S \circ \mathbf{Z}_{S^c}) = h(\mathbf{Z})] = 2^{-\sum_{i \in S} b_i},$$

since each h_j is uniformly random and since each row of $\mathbf{Z}_S$ differs from the corresponding row of $\mathbf{Z}$. Thus, we may bound

$$\begin{aligned}
\Pr[\mathcal{E}_S] &\leq \sum_{\tilde{\mathbf{Z}}_S \in \mathcal{Z}_S^\ell \setminus \{\mathbf{Z}_S\}} \mathbf{1}[\tilde{\mathbf{Z}}_S \circ \mathbf{Z}_{S^c} \in T_\varepsilon^{(\ell)}] \cdot 2^{-\sum_{i \in S} b_i} \\
&\leq \sum_{\pi \in \Delta_\ell(\mathcal{D})} \sum_{\tilde{\mathbf{Z}}_S \in \mathcal{Z}_S^\ell} \mathbf{1}[\tilde{\mathbf{Z}}_S \circ \mathbf{Z}_{S^c} \in T_{\varepsilon,\sigma_\pi}^{(\ell)}] \cdot 2^{-\sum_{i \in S} b_i} &\quad (11) \\
&\leq \sum_{\pi \in \Delta_\ell(\mathcal{D})} \left| T_{\varepsilon,\sigma_\pi,S}^{(\ell)}(\mathbf{Z}_{S^c}) \right| \cdot 2^{-\sum_{i \in S} b_i} &\quad (12) \\
&\leq \sum_{\pi \in \Delta_\ell(\mathcal{D})} (\ell+1)^{|\mathcal{Z}|} \cdot 2^{\ell H(\mathbf{z}'_S | \mathbf{z}'_{S^c})(1+\delta(\varepsilon)) - \sum_{i \in S} b_i} &\quad (13) \\
&\leq \ell^{|\mathcal{X}|^m + |\mathcal{Z}|} \max_{\pi \in \Delta(\mathcal{D})} 2^{\ell H(\mathbf{z}'_S | \mathbf{z}'_{S^c})(1+\delta(\varepsilon)) - \sum_{i \in S} b_i} &\quad (14)
\end{aligned}$$

where in (11) we have used the definition of $T_\varepsilon^{(\ell)}$, in (12) we have used Definition 13, in (14) we have used Theorem 13, and in (14) we have used the fact (8) that $|\Delta_\ell(\mathcal{D})| \leq \ell^{|\mathcal{X}|^m}$.

Thus, provided that

$$\sum_{i \in S} b_i \geq \max_{\pi \in \Delta(\mathcal{D})} \ell \left(H(\mathbf{z}'_S | \mathbf{z}'_{S^c})(1 + \delta(\varepsilon)) + \varepsilon \right) + (|\mathcal{X}|^m + |\mathcal{Z}|) \log(\ell),$$

the probability that $\mathcal{E}_S$ occurs is at most

$$\Pr[\mathcal{E}_S] \leq \exp(-\Omega(\ell \varepsilon)).$$

Using the fact that $|\mathcal{X}|^m, |\mathcal{Z}|$ are constants that depend only on the original HSS scheme Π (and not on ℓ or ε), for sufficiently large ℓ it suffices to require that

$$\sum_{i \in S} b_i \geq \max_{\pi \in \Delta(\mathcal{D})} \ell \cdot (H(\mathbf{z}'_S | \mathbf{z}'_{S^c}) + \delta'(\varepsilon))$$

for some function δ' that depends only on Π so that $\delta'(\varepsilon) \to 0$ as $\varepsilon \to 0$. This proves the claim. □

Finally, Claim 3 implies that with high probability, (B) occurs. Indeed, by a union bound over all 2^k choices for S, we conclude that

$$\begin{aligned} \Pr[\text{ (B) does not occur }] &= \Pr[\ \mathcal{E}_S \text{ occurs for some } S \subseteq [k]\ ] \\ &\leq \sum_{S \subseteq [k]} \Pr[\mathcal{E}_S] \\ &\leq 2^k \exp(-\Omega(\varepsilon \ell)) \\ &\leq \exp(-\Omega(\varepsilon \ell)), \end{aligned}$$

using the fact that k is a constant that depends only on the HSS Π. Since Claim 2 has established that (A) occurs with probability at least $1 - \exp(-\Omega(\varepsilon^2 \ell))$, we conclude by another union bound that both (A) and (B) occur with probability at least $1 - \exp(-\Omega(\varepsilon^2 \ell))$. As discussed above, if both (A) and (B) occur, then the algorithm Rec' is correct. This completes the proof of the theorem. □

5.2 Symmetrically Private HSS

In HSS, the only security guarantee is that any set of t colluding servers learn nothing about the input secrets. We can also study the stronger notion of *symmetric security*, where we additionally demand that the output client learn nothing beyond the desired output $f(x_1, \ldots, x_m)$.

In this section, we begin an exploration of symmetric security that can beat the $1 - dt/k$ barrier. First, we show that the Shamir-based non-linear HSS example at the beginning of Section 5 is *not* symmetrically secure; then we show that the Greedy Monomial CNF (Definition 12) that is used in the proof of Corollary 6 *is* symmetrically secure. We leave it as an intriguing open direction to further characterize which non-linear HSS schemes are symmetrically secure.

To begin, we first formally define symmetric security.

Definition 14 (SHSS)**.** *Let* $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ *be an HSS for* $\mathcal{F}$ *with inputs in* $\mathcal{X}^m$*. We say that* Π *is a* symmetrically private *HSS (SHSS) if the following holds for all* $f \in \mathcal{F}$ *and all* $\mathbf{x} \in \mathcal{X}^m$*. Let* $y^{(1)}, \ldots, y^{(k)}$ *denote the output shares of* Π *(that is, the outputs of* Eval *given* f*). Then the joint distribution of* $y^{(1)}, \ldots, y^{(k)}$ *depends only on* $f(\mathbf{x})$*.*

Remark 13 (Relationship to SPIR)**.** *A related notion is that of* symmetrically private information retrieval *(SPIR) [GIKM98], where the (single) client only learns its requested record from the database and nothing else. The notion of symmetric privacy in SPIR is stronger than the one we consider here in that it rules out additional information about the function* f *in the joint distribution of* both input shares and output shares*. To meet this stronger requirement, the servers must inherently share a source of common randomness which is unknown to the client. Our weaker symmetric privacy notion considers the output shares alone. This is motivated by applications in which the output shares are delivered to an external output client who does not collude with any servers or input clients.*

We first note that schemes with linear reconstruction, such as our constructions from Sections 3.1 and 4, can be trivially made into weak SHSS schemes by having some input client distribute among the k servers a random sharing of $\mathbf{0} \in \mathbb{F}^\ell$ according to the ℓ-LMSSS defined by the output shares of the HSS. Then, each servers adds its share of $\mathbf{0}$ to its output share. Unfortunately, this will result in a fresh sharing of the secret, which is not compressible, and thus we may not apply the Slepian-Wolf-like machinery from the previous section in order to beat the $1 - dt/k$ barrier.

Hence, it is interesting to ask whether we can construct HSS schemes which go beyond the $1 - dt/k$ barrier *and* are an SHSS. We begin by arguing that the usual Shamir-based HSS (instances of which can be compressed according to the warm-up in Section 5) is not an SHSS. The fact the output shares don't encode a uniformly random polynomial (because it is necessarily reducible) is well known, see for example [BGW88, CCD88]. We prove the following proposition for completeness.

Proposition 3 (Shamir-based HSS is not an SHSS)**.** *Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| > 3$. Let Π be the 1-private 3-server HSS for $\mathcal{F} = \{f(x_1, x_2) \mapsto x_1x_2\}$ where* Share *is given by Shamir sharing (Definition 7) and where* Eval *simply computes the conversion in Lemma 3. Then Π is not an SHSS.*

Proof. Let $\alpha_1, \alpha_2, \alpha_3$ be the evaluation points used in the HSS. Suppose first that we choose inputs $(x_1, x_2) = (0, 0)$. For $i \in [2]$, let $p_i \in \mathbb{F}[x]$ be the random polynomial of degree 1 such that $p_i(0) = x_i = 0$. After the servers apply $\mathsf{Eval}(f, \cdot, \cdot)$, the output shares constitute the polynomial $p = p_1 \cdot p_2$, evaluated at points $\alpha_1, \alpha_2, \alpha_3$. Since the degree of the polynomial is at most 2 we can fully recover it using 3 evaluation points. Since $p_i(0) = 0$ for every $i \in [2]$, the smallest (possibly) non-zero monomial in p has degree 2.

Now, suppose alternatively that we choose inputs $(x_1, x_2) = (0, 1)$. Now the smallest (possibly) non-zero monomial in $p = p_1p_2$ has degree 1. Since the random coefficient of this monomial is not always zero, the above distributions of p are not identical, while $0 \cdot 0 = 0 \cdot 1$, and so this HSS is not an SHSS. □

On the flip side, as it turns out, the HSS that arises from the proof of Corollary 6 is an SHSS. Given Proposition 3, this is an additional benefit that the "Greedy Monomial CNF" scheme has over Shamir-sharing, aside from its better compressibility compared to Shamir (as in Remark 11).

Proposition 4 (Greedy-CNF-based HSS is an SHSS)**.** *The HSS scheme Π from Definition 12 for $t = 1, d = 2, k = 3$ is an SHSS.*

Proof. Suppose that we want to multiply $\alpha, \beta \in \mathbb{F}$ using Π. To this end, suppose we additively share $\alpha = a_1 + a_2 + a_3$ and $\beta = b_1 + b_2 + b_3$ and, according to the 1-CNF sharing, we give server with index $j \in [3]$ the shares $(a_i, b_i)_{i \neq j}$. Then, the servers apply their Eval algorithm and obtain output shares

$$\begin{aligned} y_1 &= (a_2 + a_3)(b_2 + b_3) \\ y_2 &= a_1b_1 + a_1b_3 + a_3b_1 \\ y_3 &= a_1b_2 + a_2b_1 \end{aligned}$$

where server with index j outputs y_j. We need to show that the distribution of (y_1, y_2, y_3) depends only on $a \cdot b$. To this end, let $\alpha, \alpha', \beta, \beta' \in \mathbb{F}$ be such that $\alpha\beta = \alpha'\beta'$. In the same way (y_1, y_2, y_3) depends on (α, β), suppose that (y'_1, y'_2, y'_2) depends on (α', β'). We need to prove that

(y_1, y_2, y_3) and (y'_1, y'_2, y'_2) are identically distributed. Our strategy will be to find a bijective mapping $f_{\alpha,\alpha',\beta,\beta'}(a_1, a_2, b_1, b_2) = (a'_1, a'_2, b'_1, b'_2)$ such that if we replace a_1, a_2, b_1, b_2 by a'_1, a'_2, b'_1, b'_2 in the expression for (y_1, y_2, y_2) we get (y'_1, y'_2, y'_3). This can be shown to be equivalent to requiring that

$$\begin{aligned} a_1b_2 + a_2b_1 &= a'_1b'_2 + a'_2b'_1 \\ a_1\beta + b_1\alpha - a_1b_1 &= a'_1\beta' + b'_1\alpha' - a'_1b'_1 \end{aligned}$$

The rest of the proof proceeds by case analysis.

Case 1: $\alpha\beta = \alpha'\beta' \neq 0$. We have that $\alpha, \beta, \alpha', \beta' \neq 0$ so we may choose

$$\begin{aligned} a'_1 &= a_1(\beta')^{-1}\beta \\ b'_1 &= b_1(\alpha')^{-1}\alpha \\ a'_2 &= a_2\alpha'\alpha^{-1} \\ b'_2 &= b_2\beta'\beta^{-1} \end{aligned}$$

which is a bijection.

Case 2: $\alpha\beta = \alpha'\beta' = 0$. If $\alpha = \alpha' = 0$ and $\beta, \beta' \neq 0$ then we require

$$\begin{aligned} a_1b_2 + a_2b_1 &= a'_1b'_2 + a'_2b'_1 \\ a_1\beta - a_1b_1 &= a'_1\beta' - a'_1b'_1 \end{aligned}$$

for which we may use the mapping

$$\begin{aligned} a'_1 &= a_1(\beta')^{-1}\beta \\ b'_1 &= b_1(\beta')^{-1}\beta \\ a'_2 &= a_2\beta'\beta^{-1} \\ b'_2 &= b_2\beta'\beta^{-1} \end{aligned}$$

which is a bijection. The case $\beta = \beta' = 0$ and $\alpha, \alpha' \neq 0$ is symmetric. If $\alpha = \alpha' = \beta' = 0$ and $\beta \neq 0$ we require

$$\begin{aligned} a_1b_2 + a_2b_1 &= a'_1b'_2 + a'_2b'_1 \\ a_1\beta - a_1b_1 &= -a'_1b'_1 \end{aligned}$$

for which we may use the mapping

$$\begin{aligned} a'_1 &= a_1 \\ b'_1 &= \begin{cases} b_1 + \beta, & a_1 \neq 0 \\ b_1, & a_1 = 0 \end{cases} \\ a'_2 &= a_2 \\ b'_2 &= \begin{cases} b_2 - \beta a_2(a_1)^{-1}, & a_1 \neq 0 \\ b_2, & a_1 = 0 \end{cases} \end{aligned}$$

which is also a bijection. The case $\alpha' = \beta' = \beta = 0$ and $\alpha \neq 0$ is symmetric. Also, if $\alpha = \beta = \alpha' = 0$ and $\beta' \neq 0$ we may use the inverse of the above mapping. □

Somewhat surprisingly, this pattern does not continue to $k = 4$, as we show in the following example:

Example 4. *The HSS from Definition 12 with* $t = 1, d = 3, k = 4, \mathbb{F} = \mathbb{F}_3$ *is not an SHSS.*

Proof. By direct calculation, when $(x_0, x_1, x_2) = (0, 0, 1)$ we get for the output shares $\Pr(0, 0, 0, 0) = 431/2187$, while when $(x_0, x_1, x_2) = (0, 0, 0)$ we get for the output shares $\Pr(0, 0, 0, 0) = 17/81$. □

Acknowledgements

We thank Elette Boyle and Tsachy Weissman for helpful conversations and the ITCS reviewers for useful comments.

References

[ACE+21] Mark Abspoel, Ronald Cramer, Daniel Escudero, Ivan Damgård, and Chaoping Xing. Improved single-round secure multiplication using regenerating codes. *IACR Cryptol. ePrint Arch.*, 2021:253, 2021.

[Bal12] Simeon Ball. On sets of vectors of a finite vector space in which every subset of basis size is a basis. *Journal of the European Mathematical Society*, 14(3):733–748, 2012.

[BCG+19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO*, pages 489–518, 2019.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *TCC*, 2019.

[Bei96] Amos Beimel. *Secure schemes for secret sharing and key distribution.* PhD thesis, Technion, 1996.

[Ben86] Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of A secret sharing. In Andrew M. Odlyzko, editor, *CRYPTO '86*, pages 251–260, 1986.

[BER17] Rawad Bitar and Salim El Rouayheb. Staircase codes for secret sharing with optimal communication and read overheads. *IEEE Transactions on Information Theory*, 64(2):933–943, 2017.

[BF90] Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In *STACS 90*, pages 37–48, 1990.

[BFKR90] Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Security with low communication overhead. In *CRYPTO '90*, pages 62–76, 1990.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *EUROCRYPT 2015, Part II*, pages 337–367, 2015.

[BGI16a] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 509–539. Springer, 2016.

[BGI16b] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1292–1303. ACM, 2016.

[BGI$^+$18] Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 21:1–21:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[BGKL03] László Babai, Anna Gál, Peter G. Kimmel, and Satyanarayana V. Lokam. Communication complexity of simultaneous messages. *SIAM J. Comput.*, 33(1):137–166, 2003.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, 1988.

[BIK05] Amos Beimel, Yuval Ishai, and Eyal Kushilevitz. General constructions for information-theoretic private information retrieval. *J. Comput. Syst. Sci.*, 71(2):213–247, 2005.

[BIKO12] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Ilan Orlov. Share conversion and private information retrieval. In *CCC 2012*, pages 258–268, 2012.

[BIW07] Omer Barkol, Yuval Ishai, and Enav Weinreb. On locally decodable codes, self-correctable codes, and t-private PIR. In *RANDOM 2007*, pages 311–325, 2007.

[BKS19] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In *EUROCRYPT 2019, Part II*, pages 3–33, 2019.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, 1988.

[CCG$^+$07] Hao Chen, Ronald Cramer, Shafi Goldwasser, Robbert de Haan, and Vinod Vaikuntanathan. Secure computation from random error correcting codes. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 291–310. Springer, 2007.

[CDI05] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Joe Kilian, editor, *Theory of*

Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings, volume 3378 of *Lecture Notes in Computer Science*, pages 342–362. Springer, 2005.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT*, 2000.

[CGKS98] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. *J. ACM*, 1998.

[CHJ09] Jun Chen, D-k He, and Ashish Jagmohan. The equivalence between slepian-wolf coding and channel coding under density evolution. *IEEE Transactions on Communications*, 57(9):2534–2540, 2009.

[CM21] Geoffroy Couteau and Pierre Meyer. Breaking the circuit size barrier for secure computation under quasi-polynomial LPN. In *EUROCRYPT 2021, Part II*, pages 842–870, 2021.

[CT06] Thomas Cover and Joy Thomas. *Elements of Information Theory, 2nd Edition.* Wiley, 2006.

[DGI$^+$19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO 2019, Part III*, pages 3–32, 2019.

[DGW$^+$10] Alexandros G Dimakis, P Brighten Godfrey, Yunnan Wu, Martin J Wainwright, and Kannan Ramchandran. Network coding for distributed storage systems. *IEEE transactions on information theory*, 56(9):4539–4551, 2010.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 93–122. Springer, 2016.

[DRWS11] Alexandros G Dimakis, Kannan Ramchandran, Yunnan Wu, and Changho Suh. A survey on network codes for distributed storage. *Proceedings of the IEEE*, 99(3):476–489, 2011.

[Efr09] Klim Efremenko. 3-query locally decodable codes of subexponential length. In *STOC*, 2009.

[EGKY21] Karim Eldefrawy, Nicholas Genise, Rutuja Kshirsagar, and Moti Yung. On regenerating codes and proactive secret sharing: Relationships and implications. In Colette Johnen, Elad Michael Schiller, and Stefan Schmid, editors, *Stabilization, Safety, and Security of Distributed Systems - 23rd International Symposium, SSS 2021, Virtual Event, November 17-20, 2021, Proceedings*, volume 13046 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2021.

[EH63] Paul Erdős and Haim Hanini. On a limit theorem in combinatorical analysis. *Publ. Math. Debrecen*, 10:10–13, 1963.

[FGJI17] Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith III. Homomorphic secret sharing from Paillier encryption. In *Provable Security*, 2017.

[FIKW22] Ingerid Fosli, Yuval Ishai, Victor I. Kolobov, and Mary Wootters. On the download rate of homomorphic secret sharing. In *ITCS*, 2022.

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 699–710. ACM, 1992.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

[GFC06] Javier Garcia-Frias and Felipe Cabarcas. Approaching the slepian–wolf boundary using practical channel codes. *Signal Processing*, 86(11):3096–3101, 2006.

[GH19] Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In *TCC*, 2019.

[GI99] Niv Gilboa and Yuval Ishai. Compressing cryptographic resources. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 591–608. Springer, 1999.

[GIKM98] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 151–160. ACM, 1998.

[GM10] Berndt M Gammel and Stefan Mangard. On the duality of probing and fault attacks. *Journal of electronic testing*, 26(4):483–493, 2010.

[GW16] Venkatesan Guruswami and Mary Wootters. Repairing reed-solomon codes. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 216–226. ACM, 2016.

[HB17] Wentao Huang and Jehoshua Bruck. Secret sharing with optimal decoding and repair bandwidth. In *2017 IEEE International Symposium on Information Theory, ISIT 2017, Aachen, Germany, June 25-30, 2017*, pages 1813–1817. IEEE, 2017.

[HH19] Syed Mahbub Hafiz and Ryan Henry. A bit more than a bit is more than a bit better. *Proceedings on Privacy Enhancing Technologies*, 2019(4), 2019.

[HLKB16] Wentao Huang, Michael Langberg, Jörg Kliewer, and Jehoshua Bruck. Communication efficient secret sharing. *IEEE Trans. Inf. Theory*, 62(12):7195–7206, 2016.

[ILM21] Yuval Ishai, Russell W. F. Lai, and Giulio Malavolta. A geometric approach to homomorphic secret sharing. In *PKC 2021, Part II*, pages 92–119, 2021.

[ISN89] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989.

[LMS18] Russell W. F. Lai, Giulio Malavolta, and Dominique Schröder. Homomorphic secret sharing for low degree polynomials. In *ASIACRYPT*, 2018.

[Mas95] James L Massey. Some applications of coding theory in cryptography. *Codes and Ciphers: Cryptography and Coding IV*, pages 33–47, 1995.

[Mau06] Ueli M. Maurer. Secure multi-party computation made simple. *Discret. Appl. Math.*, 154(2):370–381, 2006.

[OSY21] Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of Paillier: Homomorphic secret sharing and public-key silent OT. In *EUROCRYPT 2021, Part I*, pages 678–708, 2021.

[RAD78] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In Richard A. DeMillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, *Foundations of Secure Computation*, pages 165–179. Academic Press, 1978.

[RKV18] Ankit Singh Rawat, Onur Ozan Koyluoglu, and Sriram Vishwanath. Centralized repair of multiple node failures with applications to communication efficient secret sharing. *IEEE Transactions on Information Theory*, 64(12):7529–7550, 2018.

[RS21] Lawrence Roy and Jaspal Singh. Large message homomorphic secret sharing from DCR and applications. In *CRYPTO 2021, Part III*, pages 687–717, 2021.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 1979.

[SJ16] Hua Sun and Syed Ali Jafar. The capacity of private information retrieval. In *2016 IEEE Global Communications Conference, GLOBECOM 2016, Washington, DC, USA, December 4-8, 2016*, pages 1–6. IEEE, 2016.

[SJ17] Hua Sun and Syed Ali Jafar. Optimal download cost of private information retrieval for arbitrary message length. *IEEE Trans. Inf. Forensics Secur.*, 12(12):2920–2932, 2017.

[SJ18] Hua Sun and Syed Ali Jafar. The capacity of robust private information retrieval with colluding databases. *IEEE Trans. Inf. Theory*, 64(4):2361–2370, 2018.

[SMT15] Salman Salamatian, Muriel Médard, and Emre Telatar. A successive description property of monotone-chain polar codes for slepian-wolf coding. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 1522–1526. IEEE, 2015.

[SPDC14] Karthikeyan Shanmugam, Dimitris S Papailiopoulos, Alexandros G Dimakis, and Giuseppe Caire. A repair framework for scalar mds codes. *IEEE Journal on Selected Areas in Communications*, 32(5):998–1007, 2014.

[SRR14] Nihar B. Shah, K. V. Rashmi, and Kannan Ramchandran. One extra bit of download ensures perfectly private information retrieval. In *2014 IEEE International Symposium on Information Theory, Honolulu, HI, USA, June 29 - July 4, 2014*, pages 856–860. IEEE, 2014.

[SW21] Noah Shutty and Mary Wootters. Low-bandwidth recovery of linear functions of reed-solomon-encoded data. *arXiv preprint arXiv:2107.11847*, 2021.

[TYB17] Itzhak Tamo, Min Ye, and Alexander Barg. Optimal repair of reed-solomon codes: Achieving the cut-set bound. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 216–227. IEEE Computer Society, 2017.

[Yek07] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. In *STOC*, 2007.

A Computationally Secure HSS

In this section we define the computational relaxation of HSS, adapting earlier definitions (see, e.g., [BGI+18]) to our notation.

Unlike the information-theoretic setting of Definition 1, in the computational setting the input domain $\mathcal{X}$ and output domain $\mathcal{Y}$ are $\{0,1\}^*$ rather than finite sets. We further modify the syntax of Definition 1 in the following ways.

- The function Share takes a security parameter λ as an additional input.
- The function class $\mathcal{F}$ is replaced by a polynomial-time computable function $F(\hat{f}; x_1, \ldots, x_m)$, where $\hat{f}$ describes a function $f(x_1, \ldots, x_m)$ and is given as input to Eval. For instance, private information retrieval can be captured by $F(\hat{f}; x_1)$ where $\hat{f}$ describes an N-symbol database and x_1 an index $i \in [N]$, and F returns $\hat{f}[x_1]$. When referring to HSS for concrete computational models such as circuits or branching programs, the input $\hat{f}$ is a description of a circuit or a branching program with inputs $x_1, \ldots, x_m$. Finally, when considering *additive* HSS as in Definition 3, $\hat{f}$ also specifies the finite field over which the output is defined.

Security for computational HSS is defined in the following standard way.

Definition 15 (Computational HSS: Security)**.** *We say that* $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ *is* computationally t-private *if for every set of servers* $T \subset [k]$ *of size* t *and polynomials* p_1, p_2 *the following holds. For all input sequences* x_λ, x'_λ *such that* $|x_\lambda| = |x'_\lambda| = p_1(\lambda)$*, circuit sequences* C_λ *such that* $|C_\lambda| = p_2(\lambda)$*, and all sufficiently large* λ*, we have*

$$\Pr[C_\lambda(Y_T) = 1] - \Pr[C_\lambda(Y'_T) = 1] \leq 1/p_2(\lambda),$$

where Y_T *and* Y'_T *are the* T*-entries of* $\mathsf{Share}(1^\lambda, x_\lambda)$ *and* $\mathsf{Share}(1^\lambda, x'_\lambda)$*, respectively.*

B Proof of Proposition 1

In this section we prove Proposition 1.

First, we show that there exists a CNF-based scheme. By (the proof of) Theorem 3, it suffices to exhibit a 5-party 4-LMSSS over $\mathbb{F}_2$ with download cost 5. For secrets $x_1, x_2, x_3, x_4 \in \mathbb{F}_2$ and a random bit $r \in \mathbb{F}_2$, such a scheme is given by

$$\mathsf{Share}(x_1, x_2, x_3, x_4, r) = (r, x_1 + r, x_2 + r, x_3 + r, x_4 + r)$$

and

$$\mathsf{Rec}(y_1, y_2, y_3, y_4, y_5) = (y_1 + y_2, y_1 + y_3, y_1 + y_4, y_1 + y_5).$$

Next, we show that there is no Shamir-based scheme that downloads only one bit from each party. The proof proceeds by a computer search. However, naively such a search (for example, over all sets of $k = 5$ linear functions from $\mathbb{F}_8^4 \to \mathbb{F}_2$) is not computationally tractable. Instead we first analytically reduce the problem to one that is tractable.

Suppose (towards a contradiction) that there were an $\mathbb{F}_2$-linear HSS-for-concatenation Π based on Shamir-sharing. We note that since Rec is $\mathbb{F}_2$-linear, we may assume that Eval is also $\mathbb{F}_2$-linear.[17] We will write down Π linear-algebraically over $\mathbb{F}_2$. Fix some basis for $\mathbb{F}_8$ over $\mathbb{F}_2$. For any element $\alpha \in \mathbb{F}_8$, let $\mathrm{Vec}(\alpha) \in \mathbb{F}_2^3$ denote the vector representation of α in that basis, and let $\mathrm{Mat}(\alpha) \in \mathbb{F}_2^{3\times 3}$ denote the matrix representation of α. Thus, $\mathrm{Mat}(\alpha)\mathrm{Vec}(\beta) = \mathrm{Vec}(\alpha\beta)$ for all $\alpha, \beta \in \mathbb{F}_8$.

Let $x^{(j)} \in \mathbb{F}_2$ for $j \in [4]$ be the $\ell = 4$ secrets. Choose $\alpha_0, \alpha_1, \ldots, \alpha_5 \in \mathbb{F}_8^*$, and suppose without loss of generality that $\alpha_0 = 0$.[18] Under Shamir sharing, party i receives

$$(x^{(j)} + \rho^{(j)} \cdot \alpha_i)_{j \in [4]},$$

where $\rho^{(j)} \in \mathbb{F}_8$ is uniformly random. Writing this over $\mathbb{F}_2$, party i receives

$$[\mathbf{v}|\mathrm{Mat}(\alpha_i)] \cdot \begin{pmatrix} x^{(j)} \\ r_1^{(j)} \\ r_2^{(j)} \\ r_3^{(j)} \end{pmatrix} \qquad \text{for } j \in [4],$$

where $\mathbf{v} = \mathrm{Vec}(1) \in \mathbb{F}_2^3$ is a column vector, and where $r_i^{(j)} \in \mathbb{F}_2$ are uniformly random. (That is, $\mathrm{Vec}(\rho^{(j)}) = \mathbf{r}^{(j)}$). Let $W_i = [\mathbf{v}|\mathrm{Mat}(\alpha_i)]$ for all $i \in [5]$.

By assumption, each party i sends a single bit, which must be an $\mathbb{F}_2$-linear combination of the bits that they hold. Thus, each party i sends

$$z_i = \sum_{j=1}^{4} \langle \mathbf{w}_i^{(j)}, (x^{(j)}, r_1^{(j)}, r_2^{(j)}, r_3^{(j)}) \rangle,$$

[17] Indeed, by the same argument in the proof of Theorem 2, our $\mathbb{F}_2$-linear HSS-for-concatenation $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Rec})$ gives rise to an $\mathbb{F}_2$-linear 4-LMSSS that has the output shares of Π as its shares, with the same Rec function. Since Rec is $\mathbb{F}_2$-linear, and since by [Bei96] any LMSSS $\mathcal{L}$ with linear Rec may also have a $\mathbb{F}_2$-linear share function, we may assume that $\mathcal{L}$ has a $\mathbb{F}_2$-linear share function, Share'. But since Share (which is Shamir sharing) is also $\mathbb{F}_2$-linear, this implies that we may take Eval to be the $\mathbb{F}_2$-linear function $\mathsf{Share}' \circ \mathsf{Share}^{-1}$, where Share^{-1} denotes an arbitrary linear function that returns the inputs given all of the Shamir shares.

[18] Indeed, this is without loss of generality, as the Shamir scheme with general α_i has the same share function as the Shamir scheme with evaluation points α'_i where $\alpha'_0 = 0$ and $\alpha'_i = \alpha_i - \alpha_0$.

for some vectors $\mathbf{w}_i^{(j)}$ in the rowspan of W_i. Let

$$\mathbf{w}_i = \mathbf{w}_i^{(1)} \circ \mathbf{w}_i^{(2)} \circ \mathbf{w}_i^{(3)} \circ \mathbf{w}_i^{(4)} \in \mathbb{F}_2^{16},$$

where $\circ$ denotes concatenation. Since the recovery algorithm must also be linear, we have

$$x^{(j)} = \sum_{i=1}^{5} a_{ij} z_i$$

for some coefficients $a_{ij} \in \mathbb{F}_2$ and for all $j \in [4]$. This implies that for all $j \in [4]$,

$$\sum_{i=1}^{5} a_{ij} \mathbf{w}_i = \mathbf{e}_{4j+1}, \tag{15}$$

where $\mathbf{e}_r \in \mathbb{F}_2^{16}$ denotes the r'th standard basis vector. (This is because if we concatenate the vectors $(x^{(j)}, r_1^{(j)}, r_2^{(j)}, r_3^{(j)})$ to mirror the concatenation that created the $\mathbf{w}_i$'s, the $x^{(j)}$ term appears in the $4j+1$'st coordinate). Consider the restriction $\mathbf{y}_i$ of $\mathbf{w}_i$ to the coordinates indexed by $(2, 3, 4, 6, 7, 8, 10, 11, 12, 14, 15, 16)$; that is, all of the elements of $[16]$ that are not equal to $4j+1$. Then, (15) implies that for all $j \in [4]$,

$$\sum_{i=1}^{5} a_{ij} \mathbf{y}_i = \mathbf{0}.$$

The 5×4 matrix formed by the a_{ij} must be full rank, since the right hand sides of (15) for $j \in [4]$ are linearly independent. Thus, the matrix with the $\mathbf{y}_i$ as columns has a kernel of dimension at least 4. We conclude that the $\mathbf{y}_i$ for $i \in [5]$ must have a span of dimension at most 1. In other words, without loss of generality we may assume that $\mathbf{y}_i = \mathbf{y}$ is independent of i. (Notice that it's also possible that $\mathbf{y}_i = \mathbf{0}$, but in that case we could just make the corresponding a_{ij} values zero and set $\mathbf{y}_i = \mathbf{y}$).

Further, values for $\mathbf{y}_i$ determine all of the values in $\mathbf{w}_i$. This is because we have the restriction that each $\mathbf{w}_i^{(j)}$ lies in the rowspan of W_i, and $\mathrm{Mat}(\alpha_i)$ is full rank. This implies that the the first entry of $\mathbf{w}_i^{(j)}$ is given by $\langle \mathbf{y}_i^{(j)} \mathrm{Mat}(\alpha_i)^{-1}, \mathbf{v} \rangle$ for all i, j.

Therefore, the values of $\mathbf{w}_i$ for all $i \in [5]$ are determined by a single choice of $\mathbf{y} \in \mathbb{F}_2^{12}$. This suggests an algorithm to enumerate over all linear HSS with the claimed properties:

- For each choice of distinct $\alpha_1, \ldots, \alpha_5 \in \mathbb{F}_8^*$, and for each choice of $\mathbf{y} \in \mathbb{F}_2^4$:
  - Recover the vectors $\mathbf{w}_i$ for $i \in [5]$ that are implied by $\mathbf{y}$, as described above.
  - If it is the case that $\mathbf{e}_{3j+1}$ lies in the span of $\{\mathbf{w}_i : i \in [5]\}$ for all j, we have found a valid linear HSS; return it.
- If we have not returned, return "there is no such HSS."

This search is tractable: there are only $\binom{7}{5} \cdot 2^{12}$ things to enumerate over, and for each of these we must do some linear algebra on a 16×5 matrix to determine if the choice of $\mathbf{y}$ results in a valid scheme for the choice of evaluation points. We implemented this search, and did not find any valid schemes.[19] Therefore we conclude that no such scheme exists.

[19]The code for our implementation of this search is available at https://web.stanford.edu/~marykw/files/proof_of_no_Shamir_HSS.sage.