On the Download Rate of Homomorphic Secret Sharing

Ingerid Fosli[1]
Google, Houston, TX, USA

Yuval Ishai
Technion, Haifa, Israel

Victor I. Kolobov
Technion, Haifa, Israel

Mary Wootters
Stanford University, CA, USA

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.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives

Keywords and phrases Information-theoretic cryptography, homomorphic secret sharing, private information retrieval, secure multiparty computation, regenerating codes

Digital Object Identifier 10.4230/LIPIcs.ITCS.2022.71

Related Version *Full Version*: https://eprint.iacr.org/2021/1532 [30]

Funding *Yuval Ishai*: Research partially supported by ERC Project NTSC (742754), ISF grant 2774/20, and BSF grant 2018393.
Victor I. Kolobov: Same as Yuval Ishai.
Mary Wootters: Research partially supported by NSF grants CCF-1844628 and CCF-1814629, and by a Sloan Research Fellowship.

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

[1] Work done while at Stanford University.

13th Innovations in Theoretical Computer Science Conference (ITCS 2022).
Editor: Mark Braverman; Article No. 71; pp. 71:1–71:22
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Homomorphic Secret Sharing (HSS) [6, 10, 12] 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 [46, 33] 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 $f(x_1, \ldots, x_m)$.[2]

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 [6], finite field multiplication [5, 15, 21], and low-degree multivariate polynomials [2, 3, 17] 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 [9, 24, 10, 11, 29, 12, 13, 8, 18, 44, 47].

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 [25, 34, 14]. 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

[2] 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.

depend polynomially on the 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 the following we give a detailed overview of our main results. The theorem statements use the terminology informally defined above; see Section 2 for formal definitions. A high level overview of the proofs will be given in Section 1.2. The full proofs are deferred to the full version of this paper [30].

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 [2, 3] 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 [31] can be used to get the rate arbitrarily close to $1/d$, for sufficiently large ℓ.[3] We present two constructions that obtain a better rate, arbitrarily close to 1 (see [30, Section 3] for their full descriptions).

Our first construction is based on the highly redundant *CNF sharing* [41], 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.

[3] 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 [30, Remark 4] for more details.

▶ **Theorem 1.** *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 the function class of ℓ degree-d m-variate polynomials over $\mathbb{F}$ with download rate $\ell/(kb)$ and upload cost $k\ell m\binom{k-1}{t}\log_2|\mathbb{F}|$.*

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 [36, 20].

Our second construction uses *Shamir sharing* [49], where each input is shared by evaluating a random degree-t polynomial over an extension field of $\mathbb{F}$ at k distinct points.

▶ **Theorem 2.** *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 for the function class of ℓ degree-d m-variate polynomials over $\mathbb{F}$, 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 $1 - dt/k$.*

This 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 [30, Section 3.2] we show that for such linear HSS schemes, $1 - dt/k$ is the best rate possible, implying optimality of our schemes.

▶ **Theorem 3.** *Let t, k, d, m, ℓ be positive integers so that $m \geq d$. Let $\mathbb{F}$ be any finite field. Let Π be a t-private k-server linear HSS for the function class of ℓ degree-d m-variate polynomials over $\mathbb{F}$. 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 $1 - dt/k$.*

We compare the above two HSS schemes in [30, 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 [30, 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 [17] 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. [17] 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.

▶ **Theorem 4.** *For all integers $d, t, k, w > 0$, such that $dt < k$, 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 $1 - dt/k$.*

It is instructive to compare this application to a recent line of work on the download rate of PIR. Sun and Jafar [52, 53], following [48], have shown that the optimal download rate of 1-private PIR is $(1 - 1/k)/(1 - 1/k^N)$ (for records of length $\ell \geq k^N$). However, their positive result has $\Omega(N)$ upload cost. We get a slightly worse[4] download rate of $1 - d/k$, 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 [42, 40] for more examples.

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 [30, Section 4] we develop a general black-box transformation technique ([30, 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, 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 [24] (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 $\binom{k}{t}$.

▶ **Theorem 5.** *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 $\ell\binom{k-1}{t}L$.*

[4] 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 $1 - 1/k$ (where optimality follows from [48] or from [30, Lemma 3]). In Section 1.1.2 below we discuss a construction of *computationally* secure PIR that achieves the same rate of $1 - 1/k$ with *logarithmic* upload cost.

This should be compared to the $1/(t+1)$ rate obtained via a direct use of [24]. Note that, unlike recent constructions of "rate-1" fully homomorphic encryption schemes [34, 14], 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 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 [9, 11], 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 $O(\lambda \log N)$ (where λ is a security parameter).

▶ **Theorem 6.** *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 $1-1/k$.*

We can also apply this transformation to 1-private 2-server HSS schemes from [10, 44, 47], 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 [55, 26]. While this scheme is not additive, it can be made additive by doubling the number of servers. We then 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 $k_0 = 6 > 2$.

▶ **Theorem 7.** *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 $1-1/\Theta(\sqrt{k})$.*

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 [30, 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 [30, 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 [30, Theorem 12] to a 1-private 3-server HSS for AND based on CNF sharing (Definition 24), we show 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.

▶ **Corollary 8.** *For sufficiently large ℓ, the HSS from Definition 24 is a 3-server, 1-private, $\left(1 - 2^{-\Omega(\ell)}\right)$-correct HSS for the function family ℓ AND evaluations, with download rate $R \geq 0.376$.*

Moreover, for these parameters, the Greedy Monomial CNF HSS yields the best download rate when plugged into [30, Theorem 12], out of all $\mathbb{F}_2$-linear HSS schemes.

Perhaps even more surprisingly, the improved rate can be achieved while ensuring that the output shares reveal no additional information except the output. We refer to the latter feature as *symmetric privacy.*

▶ **Proposition 9.** *The 1-private 3-server HSS scheme from Definition 24 is a symmetric HSS in the sense of Definition 25.*

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 [5, 15, 21] – 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 [30, 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 [30, 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 23). 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 (see Definition 16) or Shamir sharing (Definition 17), an HSS for low-degree polynomials readily follows by exploiting the specific structure of CNF or Shamir.[5]

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 (see Corollary 21, extending [20] 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., [43, 16]). 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 (see [30, Lemma 1], generalizing [32]). This in turn leads to [30, 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 1, 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 [30, Theorem 5] for HSS-for-concatenation, and then Theorem 2 for HSS for general low-degree polynomials.

As an application of our Shamir-based construction, we extend information-theoretic PIR protocols from [17] to allow better download rate by employing more servers, while maintaining the same (sublinear) upload cost. This leads to Theorem 4.

Negative results. As mentioned above, [30, 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

[5] 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 [31]. However, this does not suffice for obtaining rate-optimal solutions for polynomials of degree $d > 1$ (see [30, Remark 4]).

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 3, 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.

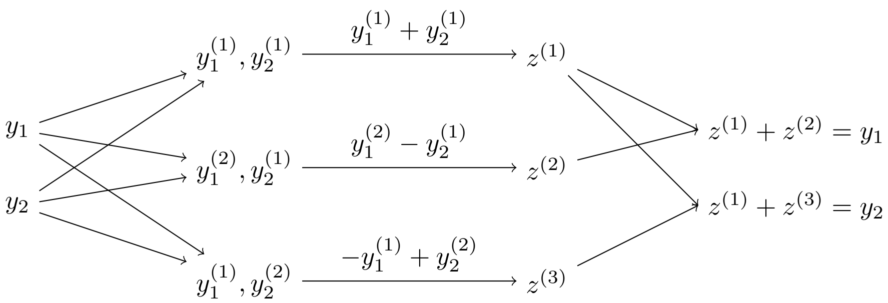
1.2.2 Black-box rate amplification for additive HSS

Next, we give a brief technical overview of our results in [30, 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 16). 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:



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 $\Theta(q^2)$ [28], 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 [30, 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., [19, 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, [30, Theorem 12]. We instantiate [30, 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 3 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 9) 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 [30, Theorem 12] in an optimal way (see Corollary 8). 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 [30, 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 $y^{(3)} = a_1b_2+a_2b_1$. 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 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 ([30, 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 [5].

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* [22], and in particular the work on using Reed-Solomon codes as regenerating codes [50, 37, 54]. 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 [22] 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 [54] (see [30, 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 [30, Remark 3]), even worse than CNF. Moreover, [54] 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 [30, 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 [54] 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;[6] 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 [37] and has ideas similar to those in [54]; 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 [1] uses regenerating codes, including a version of the scheme from [37], 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.[7] Second, the recent work [51] 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 [51] from our work is that in [51] the secrets are shared jointly, while in our setting (with several clients) the secrets must be shared independently. Thus [51] does not immediately imply any results in our setting. Finally, the work [27] studies the connection between regenerating codes and proactive secret sharing.

---

[6] 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 [23]. To the best of our knowledge, repair schemes for Reed-Solomon codes have not been studied in this setting.

[7] One may ask why [1] 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 [37], 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.

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) [39, 38, 7, 45]. 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., [39]).

However, while related, CESS – even those based on Shamir-like schemes as in [39] – 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 [39] is simply the ℓ-LMSSS described in Remark 18(b), where the ℓ inputs are interpreted as coefficients of the same polynomial and are shared *jointly*.

One exception is the scheme from [38], 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.[8]

2 Preliminaries

Notation. For an integer n, we use $[n]$ to denote the set $\{1, 2, \dots, n\}$. 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, $|w| = \log_2 |\mathcal{W}|$. 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 10** (HSS, modified from [12]). *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 $(x^{(1)}, \dots, x^{(k)})$. 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*.*
- $\mathsf{Eval}(f, j, (x_1^{(j)}, \dots, x_m^{(j)}))$*: On input $f \in \mathcal{F}$ (evaluated function), $j \in [k]$ (server index) and $x_1^{(j)}, \dots, x_m^{(j)}$ (jth share of each input), the evaluation algorithm* Eval *outputs $y^{(j)}$, corresponding to server j's share of $f(x_1, \dots, x_m)$. We refer to the $y^{(j)}$ as the* output shares*.*

[8] In more detail, the download rate of the t-private, k-server Shamir-based scheme for degree-d polynomials in [38] 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.

- $\mathsf{Rec}(y^{(1)},\dots,y^{(k)})$: *Given* $y^{(1)},\dots,y^{(k)}$ *(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* $x_1,\dots,x_m \in \mathcal{X}$ *and* $f \in \mathcal{F}$,

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

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 $\mathcal{X}^m$, $\mathbf{x}_j = (x_{1,j}, x_{2,j},\dots,x_{m,j})$ 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 10 applied to the following class $\mathcal{F}^\ell$.

▶ **Definition 11** (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*

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

Computational HSS. In this work we will be primarily interested in information-theoretic HSS as in Definition 10. However, in [30, 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 Section 2.5.

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 12** (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* $b \ge 1$; $\mathsf{Eval}(f,j,\mathbf{x}^{(j)})$ *outputs* $y^{(j)} \in \mathbb{F}^{b_j}$ *for integer* $b_j \ge 0$; *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* $\mathsf{Rec}(y^{(1)},\dots,y^{(k)}) = y^{(1)} + \ldots + y^{(k)}$.

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 13** (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* $y^{(1)},\dots,y^{(k)}$. *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 $(y^{(1)}, \dots, y^{(k)})$ reveal no additional information about the inputs beyond the output of f. Any HSS with *linear reconstruction* (Definition 12) 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 [30, Section 5].

Private information retrieval. Some of our HSS results have applications to *private information retrieval* (PIR) [17]. 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 13.

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 14** (LMSSS). *Let* $\Gamma, \mathcal{T} \subseteq 2^{[k]}$ *be monotone (increasing and decreasing, respectively)*[9] *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* $e, b_1, \dots, b_k$ *and a linear mapping* $\mathsf{Share} : \mathbb{F}^\ell \times \mathbb{F}^e \to \mathbb{F}^{b_1} \times \dots \times \mathbb{F}^{b_k}$ *so that the following holds.*

- *Correctness: For any qualified set* $Q = \{j_1, \dots, j_m\} \in \Gamma$ *there exists a linear reconstruction function* $\mathsf{Rec}_Q : \mathbb{F}^{b_{j_1}} \times \dots \times \mathbb{F}^{b_{j_m}} \to \mathbb{F}^\ell$ *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* $\ell/(b_1 + \dots + b_k)$*.*

[9] 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 $T' \in \mathcal{T}$.

Additive sharing is an important example of an LSSS.

▶ **Example 15** (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" [41] (also referred to as replicated secret sharing) and Shamir's scheme [49].

▶ **Definition 16** (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 14, which shares ℓ secrets with ℓ independent instances of CNF sharing.

▶ **Definition 17** (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 $\deg(p) \le t$ 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 18 (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 14, 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 [31]) 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 [20, 4] to multi-secret sharing.

▶ **Definition 19** (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 [20] that t-private CNF sharing can be locally converted to any LSSS $\mathcal{L}'$ which is (at least) t-private. Formally:

▶ **Theorem 20** ([20]). *Let* $\mathcal{L}$ *be the* t*-private* k*-party CNF LSSS over a finite field* $\mathbb{F}$ *(Definition 16). 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 21.** *Let* $\mathcal{L}$ *be the* k*-party* ℓ*-LMSSS given by* ℓ *instances of* t*-CNF secret sharing over* $\mathbb{F}$ *(Definition 16). 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 20 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 $\mathcal{L}'$. ◀

2.3 Linear HSS for Low-Degree Polynomials

In this section we formally define the function family of low degree polynomials, the related notion of HSS for concatenation, and a CNF-based HSS for low degree polynomials where the monomials are assigned to the servers in a greedy manner.

▶ **Definition 22.** *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 23** (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 $\mathsf{POLY}_{1,1}(\mathbb{F})^\ell$.

Finally, we define the CNF-based HSS for $\mathsf{POLY}_{d,m}(\mathbb{F})$, where the monomials are assigned to the servers in a greedy manner.

▶ **Definition 24** (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.*

2.4 Symmetric HSS

In this section we give the formal definition of a *symmetrically secure* HSS, where we additionally demand that the output client learn nothing beyond the desired output $f(x_1, \ldots, x_m)$.

▶ **Definition 25** (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 26 (Relationship to SPIR). A related notion is that of *symmetrically private information retrieval* (SPIR) [35], 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.

2.5 Computationally Secure HSS

In this section we define the computational relaxation of HSS, adapting earlier definitions (see, e.g., [12]) to our notation.

Unlike the information-theoretic setting of Definition 10, 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 10 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 12, $\hat{f}$ also specifies the finite field over which the output is defined.

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

▶ **Definition 27** (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.*

References

1. 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. URL: https://eprint.iacr.org/2021/253.
2. Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In *STACS 90*, pages 37–48, 1990.
3. Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Security with low communication overhead. In *CRYPTO '90*, pages 62–76, 1990.
4. Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Ilan Orlov. Share conversion and private information retrieval. In *CCC 2012*, pages 258–268, 2012.
5. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, 1988.
6. Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of A secret sharing. In Andrew M. Odlyzko, editor, *CRYPTO '86*, pages 251–260, 1986.
7. 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.
8. 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.
9. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *EUROCRYPT 2015, Part II*, pages 337–367, 2015.
10. 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. doi:10.1007/978-3-662-53018-4_19.

11 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. doi:10.1145/2976749.2978429.

12 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. doi:10.4230/LIPIcs.ITCS.2018.21.

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

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

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

16 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. doi:10.1007/978-3-540-72540-4_17.

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

18 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.

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

20 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. doi:10.1007/978-3-540-30576-7_19.

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

22 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.

23 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.

24 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. doi:10.1007/978-3-662-53015-3_4.

25 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.

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

27 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. doi:10.1007/978-3-030-91081-5_23.

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

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

30 Ingerid Fosli, Yuval Ishai, Victor I. Kolobov, and Mary Wootters. On the download rate of homomorphic secret sharing. *IACR Cryptol. ePrint Arch.*, 2021:1532, 2021. Full version of this paper. URL: https://eprint.iacr.org/2021/1532.

31 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. doi:10.1145/129712.129780.

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

33 Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

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

35 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. doi:10.1145/276698.276723.

36 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. doi:10.1007/3-540-48405-1_37.

37 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. doi:10.1145/2897518.2897525.

38 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. doi:10.1109/ISIT.2017.8006842.

39 Wentao Huang, Michael Langberg, Jörg Kliewer, and Jehoshua Bruck. Communication efficient secret sharing. *IEEE Trans. Inf. Theory*, 62(12):7195–7206, 2016. doi:10.1109/TIT.2016.2616144.

40 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.

41 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.

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

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

44 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.

45 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.

46 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.

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

48 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. doi:10.1109/ISIT.2014.6874954.

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

50 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.

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

52 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. doi:10.1109/GLOCOM.2016.7842315.

53 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. doi:10.1109/TIFS.2017.2725225.

54 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. doi:10.1109/FOCS.2017.28.

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