How to Recover a Secret with $O(n)$ Additions*

Benny Applebaum† Oded Nir† Benny Pinkas‡

Abstract

Threshold cryptography is typically based on the idea of secret-sharing a private-key $s \in F$ "in the exponent" of some cryptographic group G, or more generally, encoding s in some linearly homomorphic domain. In each invocation of the threshold system (e.g., for signing or decrypting) an "encoding" of the secret is being recovered and so the complexity, measured as the number of group multiplications over G, is equal to the number of F-additions that are needed to reconstruct the secret. Motivated by this scenario, we initiate the study of n-party secret-sharing schemes whose reconstruction algorithm makes a minimal number of *additions*. The complexity of existing schemes either scales linearly with $n \log |F|$ (e.g., Shamir, CACM'79) or, at least, quadratically with n independently of the size of the domain F (e.g., Cramer-Xing, EUROCRYPT '20). This leaves open the existence of a secret sharing whose recovery algorithm can be computed by performing only $O(n)$ additions.

We resolve the question in the affirmative and present such a near-threshold secret sharing scheme that provides privacy against unauthorized sets of density at most τ_p, and correctness for authorized sets of density at least τ_c, for any given arbitrarily close constants $\tau_p < \tau_c$. Reconstruction can be computed by making at most $O(n)$ additions and, in addition, (1) the share size is constant, (2) the sharing procedure also makes only $O(n)$ additions, and (3) the scheme is a blackbox secret-sharing scheme, i.e., the sharing and reconstruction algorithms work universally for all finite abelian groups F. Prior to our work, no such scheme was known even without features (1)–(3) and even for the ramp setting where τ_p and τ_c are far apart. As a by-product, we derive the first blackbox near-threshold secret-sharing scheme with linear-time sharing. We also present several concrete instantiations of our approach that seem practically efficient (e.g., for threshold discrete-log-based signatures).

Our constructions are combinatorial in nature. We combine graph-based erasure codes that support "peeling-based" decoding with a new randomness extraction method that is based on inner-product with a small-integer vector. We also introduce a general concatenation-like transform for secret-sharing schemes that allows us to arbitrarily shrink the privacy-correctness gap with a minor overhead. Our techniques enrich the secret-sharing toolbox and, in the context of blackbox secret sharing, provide a new alternative to existing number-theoretic approaches.

1 Introduction

1.1 Motivation

Threshold signatures and threshold cryptosystems [23, 22] typically rely on linear secret sharing schemes [42, 9] "over the exponent" of some abelian group $\mathbb{G}$. Specifically, each server i holds a

*This is the full version of a paper published in Crypto'23.

†Tel-Aviv University, Israel bennyap@post.tau.ac.il, odednir123@gmail.com. Supported by ISF grant no. 2805/21 and by the European Union (ERC-2022-ADG) under grant agreement no.101097959 NFITSC.

‡Aptos Labs and Bar Ilan University benny@pinkas.net

share s_i of the secret key s, and, loosely speaking, the signing process (or decryption process in threshold encryption) has the following form: The client broadcasts to the servers a public value M (e.g., the hash of the message on which we want to sign) and each server i replies with M^{s_i}. The goal of the client is to compute the signature M^s. If the client gets enough responses, say at least $\tau_c n$ out of the n servers, she can compute the signature (or decrypt the ciphertext) by computing a linear combination of the shares "in the exponent", i.e., $\prod M^{\alpha_i s_i}$, where the coefficients α_i depend on the set T of servers that are available, and satisfy $\sum \alpha_i s_i = s$. The client computes $M^{\alpha_i s_i}$ by raising each M^{s_i} to the power of α_i. By using repeated squaring, this requires between $\log|\alpha_i|$ and $2\log|\alpha_i|$ group multiplications, and so the overall complexity for $|T| = n$ is about $n\log|\alpha_i|$ multiplications.[1]

The cost of computing $\prod M^{\alpha_i s_i}$ can be quite large when there are many servers and when the group is large. For example, if one uses Shamir's secret sharing [42] the cost is $\Omega(n \log p)$ multiplications where p is the order of group $\mathbb{G}$, even without accounting for the cost of computing the Lagrange coefficients. Alternatively, by employing a blackbox secret sharing (BBSS) [23, 18] that works "universally" over any ring (or even abelian group), the cost can be made independent of p. However, the best existing schemes [20] use relatively large coefficients of bit-length $\Omega(n \log n)$ and so the recovery in the exponent takes at least $\Omega(n^2 \log n)$ multiplications. Furthermore, this holds even for the ramp setting where the correctness τ_c threshold of the scheme is bounded away from the privacy threshold τ_p. (See Section 1.5 for more details about the cost of these two approaches and of other related works.)

Our goal in this paper is to design secret-sharing schemes in which the overall complexity of the recovery is linear in the number of parties. More precisely, we would like to minimize the number of group multiplications that are performed during reconstructions. One should note that the question can (and will) be studied purely in terms of linear secret sharing schemes regardless of the encryption/signature system that is being used. Keeping in mind that every addition in $\mathbb{Z}_p$ translates into a multiplication over the group $\mathbb{G}$, we are interested in the following secret-sharing task:

> Design a secret sharing scheme over $\mathbb{Z}_p$ that supports secret-recovery with a small number of *additions*. Sepcifically, is it possible to achieve an asymptotic upper-bound of $O(n)$ additions?

1.2 Our Results

We initiate the study of Additive-Only Secret-Sharing Schemes (AOS) and settle the above question in the affirmative for *near-threshold* secret sharing schemes. Such schemes provide privacy against unauthorized sets of density at most τ_p, and correctness for authorized sets of density at least τ_c, given some arbitrarily-close constants $\tau_p < \tau_c$. We prove the following main theorem.

Theorem 1.1 (main theorem)**.** *For every constants $0 < \tau_p < \tau_c < 1$ there exists an ensemble of (τ_p, τ_c) near-threshold secret sharing schemes whose recovery algorithm makes only $O(n)$ additions. Moreover, (1) the share size is constant, (2) the sharing also makes $O(n)$ additions, and (3) the scheme is a BBSS scheme and the sharing and reconstruction algorithms work universally for all finite abelian groups $\mathbb{G}$.*

[1] The overhead of computing $\prod M^{\alpha_i s_i}$ can be reduced by computing a multi-exponentiation, namely computing the final result directly rather than computing each $M^{\alpha_i s_i}$ separately and multiplying the results. This optimization, e.g. using Pippenger's algorithm [40], improves performance by a factor of $O(\log n)$, but when $\log n \ll |\alpha_i|$ (which is the typical case in the threshold setting) this optimization has a limited effect compared to our improvements.

A few comments are in place.

- (**Ensembles**) The term *ensemble* refers to the fact that the scheme is parameterized with reusable public parameters that are sampled during the randomized set-up of the system. It is guaranteed that, except with exponentially small failure probability over the choice of the parameters, the resulting scheme satisfies correctness and privacy for all sets of density at most τ_c and τ_p, respectively. That is, each choice of the public parameters defines a scheme, and for almost all choices of the public parameters, τ_c-correctness and and τ_p-privacy hold. The public parameters can be placed in a public file and can be re-used. We can also completely remove the public parameters without affecting the asymptotic complexity of the scheme at the expense of introducing a negligible statistical error in the correctness and privacy. (See Remark 4.6.) In typical applications (e.g., threshold cryptography), this relaxation has a minor effect (if any) since the secret sharing scheme will be employed inside a computational system that can be broken anyway with a small probability.

- (**Main vs secondary features**) We view the "near-threshold" property as well as items (1–3) as "bonus" features. That is, even a weak theorem that, for every large prime p, promises a *ramp secret sharing scheme* that supports some concrete privacy and correctness thresholds (τ_p, τ_c) (e.g., $(1/3, 2/3)$) and achieves recovery with $O(n)$ additions and, say, quadratic sharing complexity and super-constant share size, would be useful for many usage scenarios. Furthermore, to the best of our knowledge, even the existence of such a weak scheme was open prior to this work. We will later present such weak versions of the main theorem that have very good concrete complexity and are likely to be useful in practice. (See Section 1.4.)

- (**Some advantages of the secondary features**) The BBSS property is especially useful for RSA-based threshold cryptography (e.g., threshold RSA signatures such as in [43]). Moreover, as a by-product, Theorem 1.1 recovers some recent fundamental results about the complexity of secret-sharing schemes, such as the existence of BBSS near-threshold schemes with constant-size shares [20] and the existence of linear-time computable secret-sharing schemes [24, 17]. (See Section 1.5). In fact, to the best of our knowledge, even if we ignore the complexity of recovery, our results are the first to obtain linear-time computable BBSS schemes for any ramp-secret sharing scheme.

- (**Application to LWE-based schemes**) Our result is also relevant in the context of LWE-based constructions (e.g., the threshold FHE of [10]). In this case, instead of placing the shares $s_i \in \mathbb{F}_p$ in the exponents, one releases "noisy" versions of the shares modulo a larger prime q, and recovery is applied over "noisy" shares. Large interpolation coefficients expand the noise magnitude by a large factor and lead to errors (e.g., bad threshold-decryption). To avoid this, one can encode separately each bit of the share, however, this means that, in the recovery, each party has to send $\log p$ noisy elements instead of a single one. Motivated by this problem, Ball et al. [2] studied the problem of secret-sharing with 0-1 reconstruction coefficients. (See Section 1.5.) We note that the binary-reconstruction requirement can be typically relaxed to the more liberal requirement of an *addition-only reconstruction algorithm with low depth* since the bit-length of the noise grows linearly with the "depth" of the algorithm. Indeed, all our constructions achieve an optimal depth of $O(\log n)$, which makes them valuable also in the LWE setting.

1.3 Technical Overview

1.3.1 Additive-only erasure codes.

We begin by ignoring the privacy condition in an attempt to construct "non-private" additive-only τ_c-correct schemes with a recovery algorithm that performs only $O(n)$ operations. When privacy is removed, this is essentially equivalent to erasure codes that correct in the presence of $(1-\tau_c)$-fraction of erasures. (For now, we think of the secret as a vector of $\Theta(n)$ field elements.) Our first observation is that graph-theoretic codes, e.g., binary low-density parity-check (LDPC) codes [26] and their derivatives (e.g., [36]) admit an additive-only decoding algorithm.

Let us focus, for concreteness, on the LDPC case. An LDPC code that maps k-long information words to n-long codewords is described by a $(n-k)\times n$ binary parity-check matrix H which is *sparse*: i.e., each of its rows/columns contains a constant number of ones. The set of codewords is the right kernel of H, i.e., all vectors $v \in \mathbb{F}_2^n$ for which $Hv = 0^{n-k}$. We think of H as a constant-degree bipartite graph G (the Tanner graph of H) whose n left vertices correspond to the codeword and its right $n-k$ nodes correspond to constraints nodes. (In other words, each constraint corresponds to a row of H.) The constraint associated with a right node asserts that the sum of all the left nodes connected to it is 0. Given a partial codeword $y_T = (y_i)_{i\in T}$ we use the following peeling-based decoding algorithm: (1) For $i \in T$ assign y_i to the ith left vertex; (2) While possible, pick a right (constraint) vertex r that all its neighbors $i_1,\ldots,i_{d-1}$ have been assigned except for one neighbor i_d and set the value of the i_dth left-node to $0-(i_1+\ldots+i_{d-1})$. (The algorithm works with any subset of $d-1$ neighbors that have already been assigned values. To simplify the notation, we denoted these neighbors as $i_1,\ldots,i_{d-1}$.) It turns out that a proper choice of the graph (or the sparse matrix) guarantees that, if one starts with a sufficiently large set of un-erased symbols T, the decoding process never stops until all the codeword is recovered. In particular, such codes can achieve a constant rate $R = k/n$ and recover from a constant fraction of erasures. (In fact, one can get an almost optimal trade-off and, for any small $\varepsilon > 0$, design a sparse LDPC that recovers the codeword from $(R+\varepsilon)$ fraction of un-erased symbol, see e.g., [37, 36].)

Observe that the above procedure works over any field, or even abelian group, $\mathbb{G}$. In particular, if the codeword is a vector that satisfies the equation $Hv = 0^{n-k}$ then the peeling-based decoder works properly.[2] Indeed, the success of the decoding is independent of the underlying domain and depends only on the *combinatorial properties* of the graph. The decoder performs at most m additions where $m = O(n)$ is the number of edges in the constant-degree graph. Let us further assume, for now, that the code also admits an encoder that maps k-long vectors to n-long codewords by making $O(n)$ blackbox additions. This assumption does not hold in general for LDPC codes, but it holds for other related codes that support similar peeling-based decoding with $O(n)$ additions, e.g., [36]. (We will also explain later how to generically rely on an arbitrary LDPC code that does not satisfy this additional requirement.)

1.3.2 From additive-only codes to additive-only secret-sharing.

It is well-known that secret-sharing schemes are closely related to erasure codes [42, 15, 38]. The literature contains two main approaches for deriving secret-sharing from codes. The first traditional approach of Massey [38] (which is also implicit in Shamir's work [42]) is algebraic in nature

[2]The condition $Hv = 0^{n-k}$ is well defined over any abelian group $\mathbb{G}$ by interpreting the multiplication of a group element by an integer as iterated addition over $\mathbb{G}$. See Section 2 for details.

and relies on the dual-distance of the code. Roughly speaking, the idea is to sample a codeword $y = (s, y_1, \ldots, y_n)$ and deliver y_i as the share of the ith party. An authorized coalition of density τ_c can recover the secret, by using decoding under $(1 - \tau_c)$-fraction of erasures. It can be proved that privacy holds for sets of density τ_p if the code has a dual distance of $\tau_p n + 1$. Unfortunately, the codes that are employed in our work (e.g., LDPC codes) fail to achieve this property.

A second, more modern, approach of Cramer et al. [17] is information-theoretic in nature. The idea is to encode a random information vector r into a codeword $y = (y_1, \ldots, y_n)$ and deliver y_i as the share of the ith party. The main observation is that, for a privacy threshold τ_p that is strictly smaller from the code's rate R, a τ_p-fraction of the parties has a small amount of information about the information vector. Specifically, given their view, the information vector is distributed uniformly over a set of size exponential in $(R - \tau_p)n$. Thus, one can use pairwise independent hashing to extract from the information word r an element that is almost-uniform conditioned on the view of the adversary, and use this element to pad the secret. (In fact, $\Omega(n)$ secrets can be packed using this approach.) One can set the parameters so that the error is sufficiently small, and apply a union-bound over all un-authorized sets. This leads to a collection of ramp (or even near-threshold) secret sharing schemes. Furthermore, if the family of hash functions is linear, the resulting scheme is linear, and if the code and hash function are computable in linear-time then so is the secret sharing scheme. Unfortunately, while there are pairwise-independent hash functions that can be computed by a linear-size arithmetic circuit [31, 4, 24] we are not aware of any pairwise-independent hash functions that can be computed by a linear number of *additions* and we conjecture that such an object does not exist.

Our approach. Our approach follows the approach of Cramer et al. [17] except that instead of applying the hash function we apply to the information vector r a random linear combination with small, constant-size integer coefficients. We replace the information-theoretic argument with a linear-algebraic argument and show that a random linear combination with small-integer coefficients "extracts" well from any source that is uniformly distributed over a "nice" low-dimensional subspace. Furthermore, this extraction works in a domain-independent way. In more detail, fix the generating matrix M of the code and a subset T, and consider a random "small" integer column vector $a \in \mathbb{N}^k$. We show that a extracts well in the following scenario: Fix an arbitrary group $\mathbb{G}$, and consider a random vector $r \xleftarrow{R} \mathbb{G}^k$, then $a \cdot r$ is almost surely uniform over $\mathbb{G}$ even conditioned on the T-restricted codeword $(Mr)_T$. Equivalently, in linear algebraic terms, for every prime p, the vector a almost surely falls out of the row span of M_T modulo p.

The actual statement depends on the magnitude of the integers in M and here the fact that the code has $O(n)$ additive complexity plays on our side. Roughly, the analysis, which uses elementary linear algebra and probability, treats separately each small prime and each prime larger than some $p_0 = O(n)$. This resembles the case analysis of BBSS of [20] with an important distinction: In [20] the authors design different schemes for each case and glue them together via CRT, and in our case the construction is uniform and the distinction between different primes happens only in the analysis. Indeed, conceptually, our approach exploits the combinatorial structure induced by graph-based codes and avoids the relatively complicated number theory that is employed by previous BBSS schemes.

1.3.3 Deriving near-threshold schemes

The techniques introduced so far yield ramp secret sharing but they fall short of providing near-threshold BBSS schemes. (The main loss is due to the analysis over small primes.) To obtain the main theorem, we import the coding-theoretic paradigm of code concatenation to the domain of secret sharing. Specifically, by using a simple combinatorial object known as *sampler graph* that satisfies some expansion-like properties, we show that it is possible to generically combine a "fast" ramp secret sharing scheme over n parties with a "slow" near-threshold scheme over a constant number d of parties, and derive a new near-threshold scheme that is almost as efficient as the fast scheme. The efficiency degrades by a constant factor that depends on the complexity of the slow scheme applied to d parties.

In our case, the fast scheme is the ramp secret sharing with $O(n)$ additive complexity for sharing and recovering from the previous section. The slow scheme can be taken to be any BBSS near-threshold or even threshold scheme, such as the scheme of Benaloh and Leichter [8] that is based on monotone formulas for the threshold function. It should be emphasized that, in our setting, the concatenation maneuver cannot be applied at the code level since the bottleneck is not the code (i.e., the correctness properties) but the analysis of the BBSS that incurs a loss in the *privacy* threshold.

We note that this amplification approach is quite generic and can be applied to any natural efficiency measure as well as to robust secret-sharing schemes. Concatenation techniques (aka party virtualization) are commonly used in the context of secure computation and distributed computing [13, 25, 29, 30, 32, 21] and also appear in protocols for verifiable secret sharing [1]. However, to the best of our knowledge, this technique was not used so far in the secret-sharing context. (Though it was implicitly used when concatenated codes were employed, e.g., by [17] who employed the code of [28].) We believe that secret-sharing concatenation forms a useful tool in the secret-sharing toolbox that is likely to lead to other applications and can probably simplify, in retrospect, previous constructions. As our concrete setting demonstrates, in some scenarios, secret-sharing concatenation cannot be replaced by concatenation in the code-level.

1.4 A Practical Instantiation

Theorem 1.1 mainly forms a feasibility result. However, our techniques give rise to potentially practical ramp secret-sharing schemes. Let us focus for concreteness on the problem of constructing threshold BLS-signatures [11] instantiated over, say, the commonly used BLS12-381 curve. In this case the signature is computed as $(H(m))^s$ in a subgroup of the curve, whose order r is a prime which is 255 bits long. The secret is s and secret sharing must be computed modulo r. In this case, we can focus on the prime $p = r$ of bit length 255 and design an LSS over the field $\mathbb{F}_p = \{0, \ldots, p-1\}$. (The following example works even for much smaller primes.) Let us assume that there are $n \geq 1000$ parties. Assume that we have an R-rate erasure code of codeword length n that can recover the information word given a fraction of $\tau_c = (R + \varepsilon_c)$ un-erased symbols, by making $D \cdot n$ additions for decoding. Then, our basic construction (encode + extract via short inner-product) can be set to have a privacy threshold of $\tau_p = (R - \varepsilon_p)$, except with statistical failure probability of 2^{-100}, so that recovering a secret costs $(D + (1.1/\varepsilon_p)n$ additions. (See Theorem 3.6.) For $\varepsilon_p = 0.1$ this adds an overhead of 11 operations per party. Next, we should decide which code to use. There are numerous options here and let us review some of them.

Using capacity-achieving LDPCs. We can use LDPC codes that almost achieve the capacity, i.e., $\tau_c = (R + \varepsilon_c)$ where ε_c can be arbitrarily small (the computational overhead D grows with $1/\varepsilon_c$). Such an ensemble of LDPC codes appears in [37, 36, 39]. However, these codes typically achieve a *weak correctness* property: For every authorized set T of density τ_c, a random code sampled from the ensemble can decode a T-partial codeword, except with probability which is inverse polynomial in the codeword length. Note that there are two issues here: (1) a non-negligible error probability and (2) the existence of "bad sets" for which decoding fails given a description of the code. The problem can be fully avoided by using other ensembles that achieve sub-optimal, yet constant, decoding capability and rate.[3] Regardless, we argue that even weakly-correct ensembles of secret-sharing schemes may be useful in some scenarios. First, observe that privacy remains "strong", that is, our construction ensures that, except for probability 2^{-100}, the scheme is private for *every* coalition of density τ_c. Keeping this in mind, we can think of the correctness threshold as a way to guarantee liveliness against random failures. In this case, the weak correctness guarantee promises that most of the time reconstruction succeeds. Furthermore, we can share the secret key independently also via some "slow" secret sharing scheme, e.g., Shamir. Whenever the fast scheme fails (due to a failure of the decoding algorithm to handle some authorized coalition T), we can use the slow track to generate a signature via Shamir's reconstruction.

Using standard LDPCs. Suppose that the privacy and correctness threshold can be far apart. A typical example is the case where $\tau_p = 1/3$ and $\tau_c = 2/3$ which corresponds to the classical setting in MPC and byzantine agreement in which the adversary can corrupt up to $n/3$ of the parties. In this case, we can use a random $(3,6)$ LDPC code whose binary parity-check matrix represents a random graph with left-degree of 3 and right degree of 6. (In fact, it is better to sub-sample the code from a sub-family of "expurgated codes"). Peeling-based decoding takes at most $3n$ additive operations and can correct up to 0.429-fraction of errors for sufficiently large lengths n (and is therefore well within the bounds $\tau_p = 1/3$ and $\tau_c = 2/3$). Concretely, for $n = 350$ (resp., 700 and 1225), decoding fails with probability smaller than 10^{-6} for erasure fractions of 0.3 (resp., 0.375 and 0.4), see [41, Figure 3.156]. The overall complexity of recovering a secret or computing a signature is less than $10n$ additions (since $\varepsilon_p = R - \tau_p = 1/2 - 1/3 = 1/6$).

Low complexity encoding. There are families of LDPC codes (and variants of them) that admit fast and even linear-time encoding (see [41]). In fact, in our context, there is a simple way to generically achieve this additional feature. Instead of generating a codeword in the Kernel of an LDPC matrix H, sample a truly random vector v and publish its syndrome $Hv = z$ as public information. The computation is fast (since H is sparse) and we can think about (H, z) as the specification of the code. The peeling-based algorithm works as before except that in each peeling step when looking at the ith constraint the right-hand side value is set to be v_i, which is whp non-zero. We abstract this idea via the notion of public shares/header and think of z as public information that is left in a public repository.

It is important to mention that this approach has a caveat. Whenever a secret is reconstructed, the public share which potentially contains large field elements should be combined in the computation (instead of only using 0's for right-hand side values). This means that a client who asks for a signature has to raise the hashed document to the power of the typically-large entries of the public

[3] In fact there are deterministic families of such codes and we employ them as part of the proof of Theorem 1.1.

share z. Still, this part of the computation can be pre-computed by the client non-interactively or while waiting for the servers' responses. Alternatively, we can partially delegate the work to the servers by asking each of them to locally raise the hashed document to the power of $(v_i : i \in S)$ for a constant-size set of indices that is determined pseudorandomly (e.g., by applying a hash function on the identity of the server). In a concrete example where $|v| = n/2$, if we ask each party to handle, say $C = 3$ random public entries, then $2n/3$ of the parties are expected to cover all but $n/2 \cdot e^{-4C/3} = n/2 \cdot e^{-12/3} \approx n/109$ of the public elements, and so the client is left with a small overhead. Of course, the whole problem can be avoided by using codes with linear additive encoding complexity (e.g., the cascade LDPC construction of [37]).

The above discussion covers only a few of the possible instantiations and other choices would likely lead to different efficiency trade-offs. We, therefore, present our constructions and proofs in a modular way that generically supports both weakly-correct ensembles and public shares/headers.

1.5 Related Work

There is a rich literature that tries to improve various efficiency measures of secret sharing schemes and most notably the share size (See Beimel's survey [6]). While we are not aware of previous works that studied the *additive complexity* of recovering secrets, let us mention some of the most relevant previous works.

Linear-time sharing. Druk and Ishai [24] constructed near-threshold linear secret sharing schemes (LSS) over constant-size fields in which one can share a secret by computing $O(n)$ arithmetic operations where multiplication is counted as a single operation. Cramer et al. [17] extended this result to the case where the secret is a vector of length $\Omega(n)$. It seems likely that these constructions generalize to larger fields $\mathbb{F}_p$. However, the recovery and sharing algorithms in this case use arbitrary field elements, which leads to $O(n \log p)$ additions. (Also note that, unlike [17], in our setting of threshold cryptography the secret is naturally interpreted as a *single element* in a large field or ring.)

BBSS schemes. BBSS schemes were first introduced by Desmedt and Frankel [23] and were further developed by [18, 19, 20] (see also references therein). Near-threshold BBSS with constant-size shares were recently constructed by [20]. Roughly speaking, they (1) glue together, via CRT, schemes that work individually for each prime $p < n$ and combine the result with (2) a scheme that works simultaneously for all large primes. Consequently, part (1) of the construction induces recovery coefficients whose order is of the order of nth primorial integer P_n (the product of the first n primes). Since the bit-length of P_n is $\Omega(n \log n)$ this means that recovering the secret under scheme (1) takes $\Omega(n^2 \log n)$ additions. Part (2) has also similar complexity since it employs "Reed-Solomon over the integers" where each entry is of magnitude $\Omega(n \log n)$, leading to $\Omega(n^2 \log n)$ addition during reconstruction.

Secret sharing with small recovery coefficients. Ball et al. [3] studied the related problem of designing threshold LSS in which the secret can be recovered by a 0-1 linear combination. Note that the existence of such a scheme with constant-size shares would also lead to recovery by $O(n)$ additions. Unfortunately, Ball et al. rule out this possibility by showing that if the recovery vector is a 0-1 vector, the share size must be $\Omega(n \log n)$ assuming that the field is of characteristic 2 or

(for general fields) assuming that the scheme satisfies a natural uniform distribution requirement. Our results bypass this lower-bound by considering the more general recovery model of $O(n)$-size additive circuit and by allowing a gap between the privacy and correctness thresholds.[4]

On the positive side, it is observed in [3] that a "bit-decomposition" of Shamir's scheme gives rise to a secret sharing scheme in which each share contains $\log|\mathbb{F}|$ field elements for $|\mathbb{F}| > n$. Reconstruction can be applied by taking a 0-1 linear combination of the shares, and so the number of additions over $\mathbb{F}$ is $\Omega(n\log|\mathbb{F}|)$ for a linear threshold of $\Theta(n)$. From our perspective, this variant has no advantage over "standard" Shamir as the total number of additions remains the same, i.e., $\Omega(n\log|\mathbb{F}|)$.

As already mentioned, for the motivating application in [3] (i.e., slow-noise-growth in LWE-based construction), the binary-reconstruction requirement can be typically relaxed to the more liberal requirement of an addition-only reconstruction algorithm with low (e.g., $\log n$) depth. From this point of view, our solutions are valuable also in the LWE setting.

Recovering a secret "in the exponent", and the cost of computing the interpolation coefficients. Although the issue we discuss here is general, we focus on the motivating example from the introduction in which a secret key $s \in \mathbb{Z}_p$ is being shared among n servers, and a client broadcasts $M \in \mathbb{G}$ and wishes to compute the value M^s. The textbook solution, based on Shamir's secret sharing, is to ask each server respond with M^{s_i}, and, assuming that the client receives the shares of a sufficiently large coalition T of size at least $\tau_c n$ recover the secret in the exponent. This process consists of two steps:

1. Computing over $\mathbb{Z}_p$ the Lagrange coefficients $(\alpha_i)_{i\in T}$ such that $s = \sum_i \alpha_i s_i$.
2. Computing $\prod M^{\alpha_i s_i}$ where the product is over $\mathbb{G}$.

The first stage is computed over $\mathbb{Z}_p$ and can be done naively by computing $O(n^2)$ additions and multiplications. It can also be implemented using FFT, as suggested in [12], and making only $O(n\log^2(n))$ additions and multiplications (and as shown in [45], this technique provides better performance, roughly from $n \approx 256$). The second stage consists of $O(n\log p)$ multiplications over the cryptographic group $\mathbb{G}$ which are typically more expensive than modular operations.

To capture the distinction between the above 2 steps, one could partition the cost of the secret-sharing recovery algorithm $\mathsf{Rec}(T, (s_i)_{i\in T\cup\{n+1\}})$ into two parts: (1) Given T the cost of generating an Addition-Only circuit Rec_T for recovering the secret given T-shares; and (2) The number of additions that Rec_T performs. Our definitional framework measures the total number of additions that are performed in both steps, and, even under this strict measure of complexity, our constructions achieve linear complexity. This is in contrast to Shamir, and all other existing schemes that achieve super-linear complexity even if we count only the complexity of Step (2).

Let us emphasize that the gain is even more significant when working over groups of unknown order. Specifically, the cost of computing Lagrange coefficients is extremely prohibitive for the case of RSA-based threshold cryptography, such as threshold RSA signatures as suggested in [43]. Indeed, since the order of the group $\mathbb{Z}_N^*$ is unknown, the interpolation coefficients must be computed over the integers. This leads to very large coefficients as they are the result of multiplying $O(n)$ integers. Subsequently, raising values to the power of these exponents modulo N can be quite inefficient.

[4] We do not know whether both relaxations are needed.

An alternative interactive solution. Deviating from the textbook solution, one can get a linear complexity via the following alternative route. The client can ask who is willing to participate in the recovery. Then, each available party broadcasts her name. Finally, once the coalition is known the coefficients are determined and each party can locally raise her share to the power of α_i and send the result to the client that just needs to multiply everything. This solution has two drawbacks: It adds interaction and it is not resilient to malicious parties or to simple failures. In contrast, our solution is non-interactive and can be easily adapted to malicious settings by assuming that the original shares are committed and that each "signature" share consists of a zero-knowledge proof of consistency. In the discrete-log setting, this can be done relatively cheaply.

2 Preliminaries

By default, all logarithms are taken to base 2. We let $h_2(\cdot)$ denote the *binary entropy function*, that maps a real number $\alpha \in (0,1)$ to $h_2(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$ and is set to zero for $\alpha \in \{0,1\}$. We use the following standard estimate for the binomial coefficients

$$\binom{n}{\alpha n} \leq 2^{h_2(\alpha)n}. \tag{1}$$

For a matrix M we let M_i and M^i denote the ith row and ith column of M, respectively. All vectors are column vectors by default. For two random variables X and Y, we say that $X \equiv Y$ if they are identically distributed.

2.1 Secret Sharing: Definitions

We begin by recalling the notion of a *partial access structures*, that defines authorized and unauthorized sets while allowing a gap between them. A *ramp access structures* is a partial access structures with two thresholds, where all sets smaller than the first threshold are unauthorized and all sets larger than the second threshold are authorized.

Definition 2.1 (partial access structure and ramp access structure)**.** *A* partial access structure *over n parties is a pair $\Gamma = (\Gamma_0, \Gamma_1)$ where $\Gamma_0, \Gamma_1 \subseteq 2^{[n]}$ are non-empty collections of sets such that $B \not\subseteq A$ for every $A \in \Gamma_0, B \in \Gamma_1$. Sets in Γ_1 are called* authorized*, and sets in Γ_0 are called* unauthorized*.*

For $0 < \tau_\mathsf{p} < \tau_\mathsf{c} \leq 1$, the $(\tau_\mathsf{p}, \tau_\mathsf{c})$-ramp access structure over n parties $\Gamma = (\Gamma_0, \Gamma_1)$ is defined by letting Γ_0 be the collection of all subsets of size at most $\tau_\mathsf{p} n$ and letting Γ_1 be the collection of all subsets of size at least $\tau_\mathsf{c} n$.

We move on and define the semantics of secret-sharing schemes. Our definition is equivalent to standard definitions (e.g., [7, 16]) though our syntax is slightly different. Notably, the dealing function, which distributes the shares, $(s_1, \ldots, s_n)$, of the secret s, is also allowed to generate an additional "public" share, s_{n+1} that is available to all parties.

Definition 2.2 (Secret-sharing schemes)**.** *A pair of deterministic algorithms, $(\mathsf{Deal}, \mathsf{Rec})$ is a* secret-sharing scheme *that realizes a (possibly partial) access structure $\Gamma = (\Gamma_0, \Gamma_1)$ with domain of secrets S, domain of random strings R, and finite domains of shares $S_1, \ldots, S_n$ and S_{n+1} (the latter domain is for public shares) if the following hold:*

- *(Correctness): For any authorized set $T \in \Gamma_1$ and every secret $s \in S$, the following* T-correctness *property holds:*

$$\Pr[\mathsf{Rec}(T, (s_i)_{i \in T \cup \{n+1\}}) \neq s] = 0,$$

  where $(s_1, \ldots, s_n, s_{n+1}) \xleftarrow{R} \mathsf{Deal}(s)$. *(The latter notation means that* r *is selected uniformly at random from* R*, and* $(s_1, \ldots, s_n, s_{n+1}) = \mathsf{Deal}(s; r)$*.)*

- *(Privacy): For any unauthorized set $T \in \Gamma_0$ and every secret $s \in S$, the following* T-privacy *property holds:*

$$(s_i)_{i \in T \cup \{n+1\}} \equiv (s'_i)_{i \in T \cup \{n+1\}}$$

  where $(s_1, \ldots, s_n, s_{n+1}) \xleftarrow{R} \mathsf{Deal}(s)$ *is a random* s*-sharing, and the vector* $(s'_1, \ldots, s'_n, s'_{n+1}) \xleftarrow{R} \mathsf{Deal}(0)$ *is a random 0-sharing for some fixed canonical element* 0 *in* S*.*

Note that privacy is a property of the sharing algorithm Deal.

Ensembles of secret sharing. Let $\Gamma = \{\Gamma_n\}_{n \in \mathbb{N}}$ be a sequence of access structures where Γ_n is an n-party (possibly partial) access structure. A triple of efficient algorithms $(\mathsf{Setup}, \mathsf{Deal}, \mathsf{Rec})$ is a δ-ensemble of Γ secret sharing schemes if for every n, except with probability $1 - \delta(n)$ over the choice of $\mathsf{pp} \xleftarrow{R} \mathsf{Setup}(1^n)$, the n-party scheme $(\mathsf{Deal}_{\mathsf{pp}}, \mathsf{Rec}_{\mathsf{pp}})$ realizes Γ_n. We highlight the following properties of this definition.

- We refer to δ as the failure or error probability of the ensemble and take it by default to be negligible in n.

- We can use a relaxed version of *ε-weakly-correct δ-private* ensemble that requires *strong privacy* and *weak correctness*. Here strong privacy means that

$$\Pr_{\mathsf{pp} \xleftarrow{R} \mathsf{Setup}(1^n)} [\forall \text{ unauthorized } T, (\mathsf{Deal}_{\mathsf{pp}}, \mathsf{Rec}_{\mathsf{pp}}) \text{ is } T\text{-private}] \geq 1 - \delta(n),$$

  and weak correctness means that for every n and every authorized set T

$$\Pr_{\mathsf{pp} \xleftarrow{R} \mathsf{Setup}(1^n)} [(\mathsf{Deal}_{\mathsf{pp}}, \mathsf{Rec}_{\mathsf{pp}}) \text{ is } T\text{-correct}] \geq 1 - \varepsilon(n).$$

  By default, we set δ to be negligible in n and set ε to some, possibly non-negligible function, that converges to 0.

- One can define weakly-private ensembles analogously though we will not use this variant in the paper. That is, in this paper even for a ε weakly-correct ensemble, except with probability δ over the choice of the public parameters, privacy holds over all unauthorized sets.

All the following variants of secret sharing (e.g., additive-only secret-sharing) can be naturally generalized to the setting of secret sharing collections. Whenever possible, we keep this extension implicit.

Remark 2.3 (Boosting weakly-correct ensembles)**.** *It is possible to reduce the correctness error ε of any weakly-correct ensemble to a negligible error via standard repetition (e.g., exponential in n) by independently sampling $k = O(n/\log(\varepsilon))$ public parameters $\mathsf{pp}_1, \ldots, \mathsf{pp}_k$ and sharing the secret k times independently with respect to each public parameter pp_i. As long as ε is polynomially-bounded away from 1, i.e., $\varepsilon < 1 - 1/\mathrm{poly}(n)$, the overhead k is polynomial and so the privacy error remains negligible. Recovery can be achieved by applying the original recovery algorithm to each part until we find an instance for which recovery succeeds.*[5] *This increases the sharing complexity and share size by a factor of k, however we can keep the expected running time of recovery essentially unchanged by applying the original recovery algorithm on the ith copy for a randomly chosen $i \in [k]$ and re-try if recovery fails.*

Remark 2.4 (deterministic constructions, public parameters and public shares)**.** *When the* Setup *algorithm is deterministic, the ensemble is referred to simply as a secret-sharing scheme. We note that one can always turn an ensemble to a deterministic construction (with statistical error) by pushing the public parameters as part of the public share. However, there is a conceptual difference between the public parameters and public share since the former can be sampled once and for all (and re-used over repeated applications) whereas the latter should be freshly sampled together with secret.*

2.2 Additive-Only Algorithms and BBSS

An additive algorithm A is an algorithm that receives two types of inputs, arithmetic data inputs $x = (x_1, \ldots, x_k)$ and some binary meta-data information $T = (T_1, \ldots, T_m)$. The algorithm A manipulates the arithmetic data by making queries to an addition/subtraction oracle that takes two arithmetic elements and returns their sum/difference. The binary meta-information can be manipulated arbitrarily. The algorithm generates arithmetic outputs $y = (y_1, \ldots, y_\ell)$. In principle, we can allow also binary outputs though we will not need this extension in this paper. The additive complexity of A is the maximal number of additions and subtractions that it performs.[6] For simplicity, we will ignore the complexity of non-arithmetic operations. Indeed, in all our constructions, the arithmetic complexity dominates the binary complexity. For any fixing T of the binary inputs and any fixing of an Abelian group $\mathbb{G}$, the algorithm $A_T^{\mathbb{G}}(x) = A^{\mathbb{G}}(T, x)$ defines a mapping from $x \in \mathbb{G}^k$ to $y \in \mathbb{G}^\ell$. This mapping can be always described by an $\ell \times k$ integer matrix M such that, for every $i \in [m]$, it holds that $y_i = \sum_{j=1}^{k} M_{i,j} x_j$ where for a positive integer k (resp., negative integer) and group element $g \in \mathbb{G}$ we write $k \cdot g$ for k-iterated additions (resp., subtractions) of g and when $k = 0$ we let $k \cdot g$ be the neutral element of $\mathbb{G}$ which will be denoted by 0.

Additive-Only Secret sharing: Syntax. We say that an (ensemble of) secret-sharing schemes is *Additive-Only* if both the distribution and recovery algorithms are additive-only algorithms. For the recovery algorithm Rec the arithmetic inputs are the shares and the binary inputs are the public parameters pp and the set of parties T that will be represented by an n-bit vector. For the distribution algorithm Deal, the vector of random elements $r = (r_1, \ldots, r_k)$ and the secret s

[5] Here we assume that given a pp, T and the shares of a T-subset, one can efficiently check whether the reconstruction succeeds or fail. This assumption always hold for linear schemes (since detecting a failure boils down to checking whether a system of equation is solvable) which are the main focus of this paper. It can also be enforced for general schemes with a relatively minor cost via standard authentication techniques.

[6] One can always reduce the number of subtractions to 1 at the expense of doubling the number of addition by maintaining for each intermediate arithmetic value v a pair of values a, b such that $v = a - b$ and postpone the actual subtraction to the end. See [44, proof of Thm 2.11] for a similar statement for the case of division/multiplication operations.

are treated as arithmetic inputs and the public-parameters pp are treated as binary inputs. As a result, the distribution algorithm can be always represented by an integer *distribution matrix* M whose rows are labeled by indices in $[n]$ such that the rows that are labeled by i correspond to the computation of the shares of the ith party. That is, for secret s and randomness $r = (r_1, \ldots, r_k)$, the share that the ith party gets is all the entries $M_j \cdot \binom{s}{r}$ for which the row j is labeled by i. Throughout the paper, we will always assume that each party gets a single group element as a share and so we may assume that the ith share is computed by the ith row of M. We assume that the public share consists of ℓ group elements and is computed by the last ℓ rows of M so the distribution matrix is always an $(n+\ell) \times (1+k)$ integer matrix.

Additive-Only Secret sharing: Semantics. We say that an additive-only secret-sharing scheme (AOS) $(\mathsf{Deal}, \mathsf{Rec})$ realizes an access structure Γ over an Abelian group $\mathbb{G}$ if $(\mathsf{Deal}^{\mathbb{G}}, \mathsf{Rec}^{\mathbb{G}})$ realizes Γ. We say that $(\mathsf{Deal}, \mathsf{Rec})$ is a black-box secret-sharing (BBSS) for Γ if $(\mathsf{Deal}^{\mathbb{G}}, \mathsf{Rec}^{\mathbb{G}})$ realizes Γ for *every* Abelian group $\mathbb{G}$. We say that an additive-only distribution algorithm Deal realizes Γ over $\mathbb{G}$ if there exists an additive-only reconstruction algorithm Rec such that $(\mathsf{Deal}, \mathsf{Rec})$ realize Γ over $\mathbb{G}$. Similarly, we say that Deal is a BBSS for Γ if there exists an additive-only reconstruction algorithm Rec such that $(\mathsf{Deal}, \mathsf{Rec})$ form a BBSS for Γ.

The following proposition follows from pioneering works about linear secret sharing and black-box secret sharing [33, 5, 18] and relates the correctness and privacy properties of an AOS to the properties of the distribution algorithm, and, more specifically, to the linear-algebraic properties of the distribution matrix M.

Proposition 2.5 (implicit in [33, 5, 18])**.** *An additive distribution algorithm* Deal *with an* $(n+\ell) \times k$ *integer distribution matrix* M *realizes an access structure* Γ *over* $\mathbb{F}_p$ *for a prime* p *if and only if:*

- *(correctness) For every authorized set* $T \subset [n]$*, the unit vector* $e_1 = (10^{n-1})$ *is spanned, modulo* p*, by the rows of matrix* M_T *that contain all the rows* $M_i, i \in [T]$ *and all the "public-share" rows* $(M_{n+1}, \ldots, M_\ell)$*.*
- *(privacy) For every unauthorized set* $T \subset [n]$*, the unit vector* $e_1 = (10^{n-1})$ *is not spanned, modulo* p*, by the rows of matrix* M_T *that contains all the rows* $M_i, i \in T$ *and all the "public-share" rows* $(M_{n+1}, \ldots, M_\ell)$*. Equivalently, there exists a* sweeping *(column) vector* $v_T \in \mathbb{N}^n$ *such that* $M_T v_T = 0 \pmod{p}$ *but* $\langle e_1, v_T \rangle \neq 0 \pmod{p}$*.*

Moreover, Deal *is a BBSS for* Γ *if the above holds for* every *prime* p*.*

The first part follows from the works of Karchmer and Wigderson [33] and Beimel [5] about the relation between linear secret sharing schemes over finite fields and span programs. The "Moreover" part implicitly follows from the work of Cramer and Fehr [18], and specifically, from Lemma 1 that asserts that an integer system over the integers is solvable if and only if it is solvable over any prime. (Indeed both the privacy and correctness conditions boils down to the solvability of an integer linear system).

2.3 Additive-Only Erasure Codes

A pair of deterministic encoder and decoder algorithms $(\mathsf{Enc}, \mathsf{Dec})$ forms an (n,k) erasure code over alphabet Σ with correctness capability of η erasures if Enc maps an information word in Σ^k

to a codeword in Σ^n, and for every $(1-\eta)n$-subset T, and every information word $x \in \Sigma^k$ and T-partial codeword $y_T = (\mathsf{Enc}(x)_i)_{i\in T}$ the decoder $\mathsf{Dec}(T, y_T)$ returns x. We consider a non-standard notion of *codes with header* in which $\mathsf{Enc}(x)$ also outputs a public header $z \in \Sigma^m$ that is always fully available to the decoder and is *not* subject to erasures. (Jumping ahead public headers correspond to public shares.) If one stores the entire information word in the public header then erasures are trivial to correct. To avoid such trivialities, we require that the length of the information word, k, will be larger than the length m of the header. Furthermore, letting $\mu := m/n$ be the *header rate*, we define the rate R of the scheme to be $\frac{k}{n} - \mu = \frac{k-m}{n}$ and require a positive rate R. This definition matches the standard definition of rate when there is no public header (i.e., $m, \mu = 0$).

We say that $(\mathsf{Enc}, \mathsf{Dec})$ are additive-only erasure codes if the algorithms are additive-only algorithms, i.e., the information word, codeword and public header are all treated as vectors of arithmetic elements over a general abelian group $\mathbb{G}$ and the set T is treated as a binary input. We require that correctness holds over any instantiation of an abelian group $\mathbb{G}$.

Ensembles of erasure codes are defined in the natural way. A tuple $\mathscr{C} = (\mathsf{CG}, \mathsf{Enc}, \mathsf{Dec})$ is a δ-ensemble of additive erasure codes that corrects up to η erasures with a rate of R and public-information rate μ if, except with probability $1-\delta$ over the choice of $\mathsf{cp} \xleftarrow{R} \mathsf{CG}(1^n)$, it holds that $(\mathsf{Enc}_{\mathsf{cp}}, \mathsf{Dec}_{\mathsf{cp}})$ form an additive erasure-code with $k = (R+\mu)n$ long information word, n-long codewords, and μn-long public header, that corrects up to η erasures. The ensemble is δ-weakly correct if for every n and $(1-\eta)n$-subset T, except with probability $1-\delta$ over $\mathsf{cp} \xleftarrow{R} \mathsf{CG}(1^n)$, it holds that for every information word $x \in \Sigma^k$ and codeword $(y,z) = (\mathsf{Enc}(x))$ the decoder $\mathsf{Dec}(T, (y_i)_{i\in T}, z)$ returns x.

3 The Basic Construction

In the basic construction, the vector of shares is sampled as a random information word r, and the ith party receives the ith entry of the corresponding codeword. In addition, we sample a random vector a of coefficients that is as long as the information word. The public information includes the public part of the codeword (if it exists), the vector a, and the value $z_0 = s + \sum_i a_i r_i$, where s is the secret that is shared. If the information word r is reconstructed, then the secret s can be recovered from z_0. We also show below that an unauthorized set learns no information about the secret.

Construction 3.1 (from additive codes to AOS)**.** *Let $\mathscr{C} = (\mathsf{CG}, \mathsf{Enc}, \mathsf{Dec})$ be a (possibly weak) δ-ensemble of additive erasure codes that corrects a fraction of $1-\tau_{\mathsf{c}}$ erasures with a rate of R and public-information rate μ. For a parameter $c \in \mathbb{N}$, define the AOS $\mathsf{Sharing} = (\mathsf{Setup}, \mathsf{Deal}, \mathsf{Rec})$ as follows:*

1. $\mathsf{Setup}(1^n)$*: Sample a code parameters via $\mathsf{cp} \xleftarrow{R} \mathsf{CG}(1^n)$ and sample an integer vector $a \xleftarrow{R} \{0,\ldots,c-1\}^{(R+\mu)n}$, output $\mathsf{pp} = (1^n, \mathsf{cp}, a)$.*
2. $\mathsf{Deal}_{\mathsf{pp}}(s)$*: Sample a random information codeword $y \in \mathbb{G}^n$ and a (possibly empty) public header $z_1 \in \mathbb{G}^{\mu n}$ by computing $(y, z_1) = \mathsf{Enc}_{\mathsf{cp}}(r)$ where $r \xleftarrow{R} \mathbb{G}^{(R+\mu)n}$ is a random information word. Set y_i to the share of the ith party, compute $z_0 = s + \sum_i a_i r_i$, and set the public share $z = (z_0, z_1)$.*
3. $\mathsf{Rec}_{\mathsf{pp}}(T, z, y_T)$*: For a set T of size at most $\tau_{\mathsf{c}} n$, recover the information word $r \in \mathbb{G}^n$ via the decoding-under-erasure algorithm $\mathsf{Dec}_{\mathsf{cp}}(z_1, T, y_T)$, and output $s = z_0 - \sum_i a_i r_i$.*

Example 3.2. *Suppose that we wish to provide n shares, and use a code that has rate $2/3$ and has no public header. Then, using the notation of Section 2.3 we have that the length of the information word is $k = 2n/3$,*

the header is of length $m = 0$, and the header rate is $\mu = 0$. The construction chooses in the setup phase a vector a of $(R+\mu)n = 2n/3$ coefficients (small integers in $\{0,\ldots,c-1\}$). In the deal phase, it chooses a random information vector r of length $(R+\mu)n = 2n/3$, sets $y = r$ and sets the public header z_1 the be empty. The recovery phase uses the code to recover r and output $s = z_0 - \sum_{i=1}^{n} a_i r_i$.

Next, consider a code with a public header that is obtained by taking the $n/3 \times n$ parity check-matrix M of the previous code, and encodes an n-long vector r by a codeword $y = r$ and a public header $z_1 = Mr$. Then, the length of the information word is $k = n$, the header is of length $m = n/3$, the header rate is $\mu = 1/3$, and the rate R remains as before, i.e., $R = k/n - \mu = 2/3$. In this case, the construction chooses in the setup phase a vector a of $(R+\mu)n = n$ coefficients. In the deal phase, it chooses a random vector r of length $(R+\mu)n = n$, sets $y = r$ and sets the public header z_1 of length $n/3$ to be the result of multiplying the generating matrix of the code by r. The recovery phase uses the code to recover r and output $s = z_0 - \sum_{i=1}^{n} a_i r_i$.

Analysis. The correctness of the scheme (over any group) follows trivially from the correctness of the erasure code. Also, if the code ensemble is only weakly-correct then so is the resulting secret-sharing ensemble. The additive complexity of sharing/recovering is exactly the complexity of encoding/decoding plus the complexity of recovering the secret from z_0, which is $(R+\mu)n\lceil\log(c-1)\rceil + (R+\mu)n \le n\lceil\log(c-1)\rceil + n$. (The inequality holds since, by assumption, $R+\mu \le 1$.) The parallel additive complexity is $\log\log c + \log n + \log(R+\mu) + 1 \le \log\log c + \log n + 1$. The next subsections (Sections 3.1 and 3.2) will be devoted to the privacy analysis and will focus on the case where the scheme is applied over $\mathbb{F}_p$ for a fixed prime.

3.1 Privacy Lemmas

To analyze the privacy of the scheme, we restrict our attention to a fixed prime and make use of the following simple claims.

Claim 3.3 (privacy from linear independence)**.** *Fix the code parameters cp, a prime p and a set $T \subset [n]$. Let G denote the generating matrix of the code $\mathsf{Enc}_{\mathsf{cp}}$ and let G_T denote the sub-matrix of G that is obtained by keeping the rows that are indexed by the set T and the "public" part. Then, the secret-sharing scheme from Construction 3.1 indexed by $\mathsf{pp} = (1^n, \mathsf{cp}, a)$ is T-private over $\mathbb{F}_p$ if and only if the vector a is not in the row-span of G_T over $\mathbb{F}_p$.*

Proof. Let G be the $(n+\mu n) \times (Rn+\mu n)$ generating matrix of the code $\mathsf{Enc}_{\mathsf{cp}}$ whose last μn rows correspond to the public header of the encoding. The distribution matrix M of the scheme is the $((1+\mu)n+1) \times (1+(R+\mu)n)$ integer matrix $M = \begin{pmatrix} 0^{(1+\mu)n} & G \\ 1 & a \end{pmatrix}$. Let M_T denote the sub-matrix of M that contains the rows that are indexed by T and the last $\mu n + 1$ rows (that corresponds to the public share). By Proposition 2.5, privacy for a set T over $\mathbb{F}_p$ is equivalent to the requirement that M_T does not span the vector e_1 over $\mathbb{F}_p$ which happens, in our case, if and only if G_T does not span the vector a. □

The following claim shows that a random small-integer vector is likely to fall out of the span of a degenerate matrix.

Claim 3.4. *Let M be an $k \times \ell$ integer matrix with $k < \ell$, let p be a prime and c be a positive integer. Then the probability that a randomly chosen vector $a \in \{0,\ldots,c-1\}^{\ell}$ is in the row-span of M computed modulo p, is at most $\alpha_{c,p}^{\ell-k} = 2^{-(\log(1/\alpha_{c,p}))(\ell-k)}$ where $\alpha_{c,p}$ equals to $1/c$ if $c \le p$ and to $\lceil c/p \rceil / c$ otherwise.*

Note that $\alpha_{c,p} < 1/c + 1/p$ for every c and p.

Proof. Denote the row span (modulo p) of M by V. The matrix M has $k' \leq k$ independent columns $M^{i_1}, \ldots, M^{i_{k'}}$ over $\mathbb{F}_p$. Let $I = \{i_1, \ldots, i_{k'}\} \subset [\ell]$ denote the set of positions of these columns. Then, for every $j \notin I$, we can write the jth column M^j as $M^I \alpha_j$ for some coefficient (row) vector $\alpha_j \in \mathbb{F}_p^{k'}$, and therefore, every vector $v \in V$ is fully determined by its values over the I-coordinates, i.e., $v_j = \alpha_j \cdot v_I$ for every index $j \notin I$. Hence, we can write $\Pr_a[a \in V]$ as

$$\begin{aligned}\sum_{v \in V} \Pr_a[a = v] &= \sum_{v \in V} \left(\Pr_a[a_I = v_I] \prod_{j \notin I} \Pr[a_j = \alpha_j \cdot v_I] \right) \\ &\leq \sum_{v \in V} \left(\Pr_a[a_I = v_I] \prod_{j \notin I} \alpha_{c,p} \right) \leq \alpha_{c,p}^{(\ell - |I|)} \leq 2^{-(\log 1/\alpha_{c,p})(\ell - k)},\end{aligned}$$

where the first equality is due to the independence of the entries of a, and the second inequality holds since, for each j and each $b \in \{0, \ldots, p-1\}$ the probability that $a_j = b \pmod p$ is at most $\alpha_{c,p}$. The claim follows. □

By combining the above claims, we derive the following lemma.

Lemma 3.5 (privacy for a fixed prime)**.** *For every code parameter* cp*, parameter* $c \in \mathbb{N}$*, and prime* p *the following holds. For every set* $T \subset [n]$ *of size* $t < Rn$*, the secret-sharing scheme from Construction 3.1 with parameters* $\mathsf{pp} = (1^n, \mathsf{cp}, a)$ *is* T*-private over* $\mathbb{F}_p$ *with probability of* $2^{-(\log \alpha_{c,p}^{-1})(Rn-t)} = (\alpha_{c,p})^{Rn-t}$ *over the choice of* a*. Furthermore, for* τ_{p} *that satisfies* $h_2(\tau_{\mathsf{p}}) < (R - \tau_{\mathsf{p}}) \log \alpha_{c,p}^{-1}$*, the scheme is* τ_{p}*-private over* $\mathbb{F}_p$ *except with exponentially small probability of* $2^{n(h_2(\tau_{\mathsf{p}}) - (R-\tau_{\mathsf{p}}) \log \alpha_{c,p}^{-1})} = 2^{-\Omega(n)}$ *over the choice of* a*.*

Proof. Fix the code parameters cp, a prime p and a set $T \subset [n]$, and let G denote the generating matrix of the code $\mathsf{Enc}_{\mathsf{cp}}$, and G_T denote the sub-matrix of G that is obtained by keeping the rows that are indexed by the set T and the "public" part (as in Claim 3.3).

The first part follows from Claims 3.4 and 3.3 by recalling that G_T is a $(t + \mu n) \times (Rn + \mu n)$ matrix for $(t + \mu n) < (Rn + \mu n)$, and the "furthermore" part follows by a union-bound over all $\tau_{\mathsf{p}} n$-size subsets of n and by using the standard inequality $\binom{n}{\tau_{\mathsf{p}} n} \leq 2^{n h_2(\tau_{\mathsf{p}})}$. □

3.2 Immediate Corollaries

Let us record two useful corollaries.

Theorem 3.6 (AOS with optimal privacy for fixed large primes)**.** *For every* $\varepsilon > 0$*, rate* $R > 0$ *and error parameter* $\beta > 0$*, take* c *to be a constant of bit length at least* $(h_2(\tau_{\mathsf{p}}) + \beta)/(R - \tau_{\mathsf{p}})$*. Then, Construction 3.1 instantiated with the constant* c *and an* δ*-ensemble (resp., weak* δ*-ensemble) of* R*-rate* $(1 - \tau_{\mathsf{c}})$*-erasure codes yields an AOS-ensemble with the following properties:*

- *(Complexity) The additive complexity (resp., parallel additive complexity) of sharing/recovering is exactly the complexity of encoding/decoding plus* $n(\lceil \log(c-1) \rceil + 1)$ *(resp.,* $\log \log c + \log n + 1$*).*
- *(Correctness) Except with probability* δ *over the choice of the code parameters* cp *and for every choice of* a*, the scheme is* τ_{c}*-correct for every prime (resp., weakly* τ_{c}*-correct if the coding ensemble is weakly correct).*

- *(Privacy) For every* $\mathsf{cp} \in \mathsf{Setup}(1^n)$ *and every prime* $p \geq c$*, except with probability* $2^{-\beta n}$ *over the choice of* a*, the algorithm* $\mathsf{Deal}_{(1^n,\mathsf{cp},a)}$ *is* $(\tau_{\mathsf{p}} = R - \varepsilon)$*-private.*

Proof. The choice of c guarantees that $\beta < \log c(R - \tau_{\mathsf{p}}) - h_2(\tau_{\mathsf{p}})$. The theorem now follows from Lemma 3.5 by recalling that $\alpha_{c,p} = 1/c$ when $p \geq c$. □

Note that the privacy threshold is almost optimal: it can be arbitrarily close to R from below. Indeed, if the underlying code is near-optimal, i.e., decoding works even when we have only $\tau_{\mathsf{c}} = (R + \varepsilon)n$ non-erased symbols for an arbitrarily small constant ε (given to the code-generation algorithm), we get a near-threshold AOS whose privacy-to-correctness gap, $\tau_{\mathsf{c}} - \tau_{\mathsf{p}}$, is 2ε. Furthermore, the concrete constants are relatively small! Assuming that the failure probability should be at most 2^{-n} (which makes sense if there are at least 100 parties), we can take $\beta = 1$ and use a constant c of bit length at most $2(R - \tau_{\mathsf{p}})^{-1}$. If the number of parties is $n \geq 1000$, then β can be taken to be 0.1 and the complexity is at most $1.1(R - \tau_{\mathsf{p}})^{-1}$.

If we do not care about the optimality of the privacy threshold, we can derive a scheme that works for every fixed prime (including small primes such as $p = 2$) with high probability.

Theorem 3.7 (AOS for small primes)**.** *For every rate* $R > 0$ *there exists small constants* τ_{p}, β *such that Construction 3.1 instantiated with any constant* $c \geq 2$ *and* δ*-ensemble (resp., weak* δ*-ensemble) of* R*-rate* $(1 - \tau_{\mathsf{c}})$*-erasure codes yields an AOS-ensemble with the following properties:*

- *Correctness and complexity as in Theorem 3.6.*
- *(Privacy) For every* $\mathsf{cp} \in \mathsf{Setup}(1^n)$ *and every prime* $p \geq 2$*, except with probability* $2^{-\beta n}$ *over the choice of* a*, the algorithm* $\mathsf{Deal}_{(1^n,\mathsf{cp},a)}$ *is* τ_{p}*-private. Consequently, the ensemble is* τ_{p}*-private simultaneously for* all *primes smaller than, say,* $2^{\beta n/2}$*, except with probability* $2^{-\beta n/2}$*.*

Proof. First, observe that for every prime p and constant $c \geq 2$ it holds that $\alpha_{c,p} < 5/6$. Indeed, if $c = 2$, it holds that $\alpha_{2,p} = 1/p \leq 1/2$ for every prime $p \geq 2$; and if $c \geq 3$, we have $\alpha_{c,p} < 1/p + 1/c \leq 1/2 + 1/3 = 5/6$ for every prime $p \geq 2$. Next, take τ_{p} to be a sufficiently small constant for which $(R - \tau_{\mathsf{p}}) \log(6/5) - h_2(\tau_{\mathsf{p}})$ is positive and take $\beta < (R - \tau_{\mathsf{p}}) \log(6/5) - h_2(\tau_{\mathsf{p}})$. This guarantees that $\beta < ((R - \tau_{\mathsf{p}}) \log \alpha_{c,p}^{-1} - h_2(\tau_{\mathsf{p}}))$ and the main part of the theorem now follows from Lemma 3.5. The "Consequently" part follows by applying a union bound over all primes of size at most $2^{\beta n/2}$. □

4 Analyzing the Basic Construction over all Primes Simultaneously

The above theorems are not sufficiently strong to handle infinitely many primes simultaneously. For this purpose, we will have to apply a more refined argument that exploits the fact that our sharing algorithm makes a linear number of additions. Let us start with the following claim that will replace Claim 3.4. In the following, we say that an integer vector $v \in \mathbb{N}^\ell$ has an *additive complexity of* e if the mapping that takes an integer vector $x \in \mathbb{N}^\ell$ to $\langle v, x \rangle$ can be computed by applying at most e additions/subtractions.

Claim 4.1. *Let* M *be an* $k \times \ell$ *integer matrix where* $k < \ell$ *and assume that each row of* M *has an additive complexity of* e*.*[7] *Let* b *be an integer and* c *be a prime of size at least* $2b$*. Then, except with failure probability* $b^{k-\ell}(2e+1)^k$ *over the choice of* $a \in \{0, \ldots, c-1\}^\ell$*,*

$$\exists \text{ integer vector } v \in [\pm b]^\ell \text{ s.t.} \quad Mv = 0^k \qquad \text{and} \qquad \langle v, a \rangle \neq 0, \tag{2}$$

[7] The hypothesis can be relaxed so that e only upper-bounds the average additive complexity of the rows in M.

where arithmetic is over the integers. Consequently, whenever (2) happens, the vector a is not in the row span of M modulo p for every prime $p > 2(c-1)b\ell$.

Proof. Let $N := b^{\ell-k}/(2e+1)^k$ and let V denote the set of integer vectors $0 \neq v \in [\pm b]^\ell$ for which the equality $Mv = 0^k$ holds over the *integers*. We begin by showing that V is of size at least $N-1$. To see this, consider the mapping ρ that takes $v \in [1..b]^\ell$ and sends it to Mv (computed over the integers). Observe that the i-th entry of Mv is an integer in the range $[-be,..,be]$ and therefore the image of ρ is of size at most $(2be+1)^k < b^k(2e+1)^k$. Since the domain of ρ is of size b^ℓ, by the pigeonhole principle, there exist a set of at least $N = b^{\ell-k}/(2e+1)^k$ distinct input vectors $v_0, \ldots, v_{N-1}$ that are all mapped to the same output. By the linearity of ρ, the $N-1$ vectors $\{v_i - v_0 | 1 \leq i \leq N-1\}$ are all non-zero vectors that are mapped by ρ to zero.

Let us further filter the set $V \subset [1-b, b-1]^\ell$ by choosing a maximal subset $V' \subset V$ of vectors that are linearly independent over $\mathbb{F}_c$. Denote the size of V' by N'. Observe that the mod-c projected set of vectors $\{v \bmod c : v \in V\}$ contains N' distinct non-zero vectors since $V \subset [1-b, b-1]^\ell$ and since the length of the interval, $[1-b, b-1]$, is smaller than $2b < c$. Therefore, $c^{N'} - 1 \geq |V|$ and $N' \geq \log_c(|V|+1) \geq \log_c(N)$.

To complete the main argument, it suffices to show that, except with probability $c^{-N'} \leq 1/N$, there exists a vector $v \in V'$ for which $\langle v, a \rangle \neq 0 \pmod{c}$. To see this, consider the random variables

$$(\langle v, a \rangle \pmod{c})_{v \in V'}$$

induced by the choice of $a \in \{0, \ldots, c-1\}^n = \mathbb{F}_c^\ell$. These random variables are mutually independent (since the vectors in V' are linearly independent) and each of them is uniformly distributed over $\mathbb{F}_c$. Therefore, the probability that all of them simultaneously take the value zero is $c^{-N'} \leq 1/N$ as promised.

Finally, the "consequently" part follows, by noting that the integer $\langle v, a \rangle$ is in the interval $[\pm(c-1)b\ell]$ and so $\langle v, a \rangle \pmod{p} \neq 0$ for every prime $p > 2(c-1)b\ell$. Since $Mv = 0^k \pmod{p}$, v certifies that the vector a is not in the row-span of M over $\mathbb{F}_p$. □

In order to employ Claim 4.1, we will need a good upper-bound e on the complexity of each row of the generating matrix of the underlying erasure code. Let us refer to such a code as e-bounded. We note that every code whose encoding can be computed by a linear number of additions can be turned into an $O(1)$-bounded code.

Claim 4.2. *There exists an efficient transformation that takes a constant $\alpha \in [0,1]$ and an additive erasure code $\mathscr{C} = (\mathsf{CG}, \mathsf{Enc}, \mathsf{Dec})$ whose additive complexity is An and outputs an (A/α)-bounded additive erasure code $\mathscr{C}' = (\mathsf{CG}, \mathsf{Enc}', \mathsf{Dec}')$ whose parameters almost match the ones of $\mathscr{C}$. Specifically, $\mathscr{C}'$ has an additive complexity of An, public rate of at least $\mu - \alpha$, rate of $R - \alpha$ and it can correct up to $\eta - \alpha$ erasures where μ, R and η are the public rate, rate and erasure correction capability of $\mathscr{C}$. Furthermore, if $\mathscr{C}$ is a (possibly weak) δ-ensemble then so is $\mathscr{C}'$.*

Proof. Fix some $\mathsf{cp} \in \mathsf{CG}(1^n)$ and observe that the average complexity of a row in the generating matrix of $\mathsf{Enc}_{\mathsf{cp}}$ is at most $\frac{An}{(1+\mu)n} < A$. Let us remove an α-fraction of the outputs whose complexity is the largest and let $\mathsf{Enc}'_{\mathsf{cp}}$ denote the resulting code. By Markov's inequality the resulting code is (A/α)-bounded. Decoding can be performed by invoking $\mathsf{Dec}_{\mathsf{cp}}$ while treating the removed entries as additional erasures. The other properties of the code can be easily verified. □

Theorem 4.3 (AOS for all large primes)**.** *For every constants $R, \mu > 0$, $e \in \mathbb{N}$, $\tau_p \in (0, R)$, error parameter $\beta > 0$, and $b \in \mathbb{N}$ whose bit-length $\log b$ is at least $(\beta + h_2(\tau_p) + (\log(e)+2)(\mu+\tau_p))/(R-\tau_p)$ the following holds. Construction 3.1 instantiated with any prime $c \geq 2b$ and δ-ensemble (resp., weak δ-ensemble) e-bounded $(1-\tau_c)$-erasure codes with rate R and public rate μ yields an AOS-ensemble with the following properties:*

- *Correctness and efficiency as in Theorem 3.6.*
- *(Privacy) For every $\mathsf{cp} \in \mathsf{Setup}(1^n)$, except with probability $2^{-\beta n}$ over the choice of a, the algorithm $\mathsf{Deal}_{(1^n,\mathsf{cp},a)}$ is simultaneously τ_p-private for* all *primes p larger than $c^2 n$ (or even $p \geq 2cb(\mu+R)n$).*

Proof. Fix some prime $c > 2b$. Fix n and code parameters $\mathsf{cp} \in \mathsf{Setup}(1^n)$ and let G denote the $(n+\mu n) \times (Rn+\mu n)$ generating matrix of the code $\mathsf{Enc}_{\mathsf{cp}}$ whose last μn rows correspond to the public header of the encoding. For a set T of size τ_p, let G_T denote the sub-matrix of G that is obtained by keeping the rows that are indexed by the set T and the "public" part (as in Claim 3.3). By Claim 3.3, it suffices to show that, except with probability $2^{-\beta n}$, the event (2) happens for G_T for every set T of size $\tau_p n$. Recall that G_T has $k = (\mu+\tau_p)n$ rows and $\ell = (R+\mu)n$ columns and each of its rows has an additive complexity of e. Hence, by Claim 4.1, the event (2) happens for any fixed T, with probability at most $b^{k-\ell}(2e+1)^k \leq 2^{-\eta n}$ where

$$\eta \geq (R-\tau_p)(\log b) - (\log(e)+2)(\mu+\tau_p).$$

When the bit length of b is larger than $(\beta + h_2(\tau) + (\log(e)+2)(\mu+\tau_p))/(R-\tau_p)$, we get that $\eta > \beta + h_2(\tau)$, which, by a union bound over all T's, yields the result. □

For example, for every fixing of constants $R > 0$, $e \in \mathbb{N}$, we can take $\tau_p \in (0, R)$ to be arbitrarily close to R (up to any small constant), and get an arbitrary exponential small error probability for all primes larger than $c^2 n$ by taking a sufficiently large constant c. Alternatively, when the code has no public information ($\mu = 0$), for every fixed odd prime c, there exist some (small) constants $\tau_p \in (0, R), \beta > 0$ for which the construction achieves τ_p-privacy except with exponentially small probability of $2^{-\beta n}$ simultaneously for all primes larger than $c^2 n$.

In any case, by combining the above theorem with Theorem 3.7 and Proposition 2.5, we derive the following theorem.

Theorem 4.4 (AOS for all primes)**.** *For every rate $R > 0$ there exists constants τ_p, β and c such that Construction 3.1 instantiated with c and δ-ensemble (resp., weak δ-ensemble) of R-rate $(1-\tau_c)$-erasure codes yields an AOS-ensemble with the following properties:*

- *Correctness and complexity as in Theorem 3.6.*
- *(Privacy) For every $\mathsf{cp} \in \mathsf{Setup}(1^n)$ and every prime $p \geq 2$, except with probability $2^{-\beta n}$ over the choice of a, the algorithm $\mathsf{Deal}_{(1^n,\mathsf{cp},a)}$ is τ_p-private simultaneously for* all *primes.*

There exists a (deterministic) construction of AOS erasure codes with linear complexity constant rate and constant erasure capability (e.g., by using the Capalbo et al. [14] unbalanced expanders in the cascade construction of Luby et al. [36]). Thus, by Proposition 2.5, we derive the following corollary.

Corollary 4.5. *For some constants $0 < \tau_p < \tau_c < 1$ there exists an (τ_p, τ_c)-ramp ensemble of BBSS scheme with constant-size shares, and where the recovery and sharing algorithms make only $O(n)$ additions, and thee setup algorithm errs with probability $2^{-\Omega(n)}$. Furthermore, the public share is a single field element and so it can be completely removed (by appending it to each party's share).*

Remark 4.6 (compressing the ensemble parameters)**.** *Since the underlying code is computed the public parameters of the secret sharing ensemble contain only the vector a whose length is $O(n)$ bits. Following [35, 17] this public information can be completely eliminated via information dispersal. Specifically, encode the vector a into a vector $A \in \{0,1\}^{O(n)}$ using some AOS-ensemble of erasure codes with linear complexity and hand A_i to the ith party. The share size remains constant and the additive complexity of sharing and recovering remains unchanged. Of course, this introduces a negligible error probability in the recovery algorithm and a negligible deviation in the privacy.*

5 Deriving Near-Threshold Schemes

We will need the following proposition about the existence of *bipartite sampler graphs*.

Proposition 5.1 (efficient samplers [34, 27])**.** *For every positive constants $\varepsilon > 0, \delta > 0$, there exists a constant $d = O(1/(\varepsilon^2\delta))$ and a $\mathrm{poly}(n, 1/\varepsilon, 1/\delta)$-time algorithm that on input 1^n outputs an bipartite graph $G = ((L,R),E)$ with the the following properties:*

1. *$|L| = |R| = n$ and the right degree of G is d.*
2. *For every $S \subset L$, it holds that for at least $1-\delta$ fraction of the vertices $r \in R$, it holds that*

$$\left| \frac{d_S(r)}{d} - \frac{|S|}{n} \right| \leq \varepsilon,$$

where $d_S(r)$ is the number of vertices in S that are connected to r.

*A graph that satisfies the above properties is called an (n,d,ε,δ)-*sampler*.*

The following lemma takes an outer "fast" ramp scheme with "bad threshold" over n parties and near-threshold "slow" scheme over a constant $d = O(1)$ number of parties, and generates a new near-optimal scheme over n parties whose complexity (sharing, reconstruction and share size) are inherited from the fast scheme up to a constant multiplicative overhead that depends on d.

Lemma 5.2. *For every ε, δ, d and n and (n,d,ε,δ)-sampler $G = ((L,R),E)$ the following holds for any $\varepsilon < \gamma < 1-\varepsilon$. Let $(\mathsf{Deal}_{\mathsf{out}}, \mathsf{Rec}_{\mathsf{out}})$ be an "outer" $(\delta, 1-\delta)$-ramp secret sharing scheme over n parties and let $(\mathsf{Deal}_{\mathsf{in}}, \mathsf{Rec}_{\mathsf{in}})$ be an "inner" $(\gamma-\varepsilon, \gamma+\varepsilon)$-ramp secret sharing scheme over d parties. Define a secret-sharing scheme $(\mathsf{Deal}, \mathsf{Rec})$ over the n parties in L as follows:*

1. *$\mathsf{Deal}(s)$: Generate n "virtual shares" $(s'_r)_{r\in R} \overset{R}{\leftarrow} \mathsf{Deal}_{\mathsf{out}}(s)$. For each $r \in R$, share each virtual share s'_r via $(s_{r,\ell})_{\ell \in L(r)} \overset{R}{\leftarrow} \mathsf{Deal}_{\mathsf{in}}(s'_r)$ where $L(r) \subset L$ is the set of left neighbors of a vertex $r \in R$ in the graph G. Set the share of the $\ell \in L$ party to be $(s_{r,\ell})_{r \in R(\ell)}$ where $R(\ell) \subset R$ is the set of right neighbors of a vertex $\ell \in L$.*

2. $\mathsf{Rec}(T,(s_i)_{i\in T})$: *For a left set $T \subset L$ denote by*

$$R_+(T) = \{r \in R : d_T(r) \geq (\gamma+\varepsilon)d\}$$

the set of right vertices r that have at least $(\gamma+\varepsilon)d$ neighbors in T. Recover all the virtual secrets (s'_r) for $r \in R_+(T)$ by applying $\mathsf{Rec}_{\mathsf{in}}((s'_{r,\ell})_{\ell\in L(r)\cap T})$ and output the result of $\mathsf{Rec}_{\mathsf{out}}((s'_r)_{r\in R_+(T)})$.

Then the secret-sharing scheme $(\mathsf{Deal},\mathsf{Rec})$ is a $(\gamma-2\varepsilon,\gamma+2\varepsilon)$-ramp secret sharing scheme.

Proof. For correctness, consider a left set T of size at least $(\gamma+2\varepsilon)n$. By the sampling properties of G, the set $R_+(T)$ is of size at least $(1-\delta)n$ and so, by the correctness of the inner scheme, at least $(1-\delta)n$ fraction of the virtual shares are recovered by the coalition T. Hence, correctness follows from the correctness of the outer secret-sharing schemes.

For privacy, consider a left set T of size at most $(\gamma-2\varepsilon)n$ and let $R_-(T) = \{r \in R : d_T(r) \leq (\gamma-\varepsilon)d\}$ be the set of right vertices r that have at most $(\gamma-\varepsilon)d$ neighbors in T. By the sampling properties of G, the set $R_-(T)$ is of size at least $(1-\delta)n$ and so, by the privacy of the inner scheme, at least $1-\delta$ fraction of the virtual shares are perfectly hidden from the coalition T. Hence, privacy follows from the privacy of the outer secret-sharing schemes. □

We note that the above lemma naturally generalizes to the case where the graph is unbalanced. (We omit the details since this variant is not needed here).

For every constant $\varepsilon > 0$, we can instantiate the lemma with an outer $(\delta,1-\delta)$-ramp AOS of linear additive complexity $O(n)$ (e.g., from Corollary 4.5) and an inner γd-threshold AOS over $d = O(1)$ parties of complexity polynomial in d (say based on the formula construction of [8]) and derive a near-threshold AOS with linear complexity as promised by the main theorem (restated here for the convenience of the reader).

Theorem 5.3 (Theorem 1.1 restated)**.** *For every constants $0 < \tau_{\mathsf{p}} < \tau_{\mathsf{c}} < 1$ there exists an ensemble of $(\tau_{\mathsf{p}},\tau_{\mathsf{c}})$ near-threshold ensemble of secret sharing schemes whose recovery algorithm makes only $O(n)$ additions. Moreover, (1) the share size is constant, (2) the sharing also makes $O(n)$ additions, and (3) the scheme is a BBSS scheme and the sharing and reconstruction algorithms work universally for all finite Abelian groups $\mathbb{G}$.*

Acknowledgements. We thank Amos Beimel for helpful discussions.

References

[1] Benny Applebaum, Eliran Kachlon, and Arpita Patra. Verifiable relation sharing and multi-verifier zero-knowledge in two rounds: Trading nizks with honest majority - (extended abstract). In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*, pages 33–56, 2022.

[2] Marshall Ball, Alper Çakan, and Tal Malkin. Linear threshold secret-sharing with binary reconstruction. In Stefano Tessaro, editor, *2nd Conference on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference*, volume 199 of *LIPIcs*, pages 12:1–12:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[3] Marshall Ball, Alper Çakan, and Tal Malkin. Linear threshold secret-sharing with binary reconstruction. In Stefano Tessaro, editor, *2nd Conference on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference*, volume 199 of *LIPIcs*, pages 12:1–12:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[4] Joshua Baron, Yuval Ishai, and Rafail Ostrovsky. On linear-size pseudorandom generators and hardcore functions. *Theor. Comput. Sci.*, 554:50–63, 2014.

[5] Amos Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis, Technion - Israel Institute of Technology, Israel, 1996.

[6] Amos Beimel. Secret-sharing schemes: A survey. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer, 2011.

[7] Amos Beimel and Benny Chor. Universally ideal secret-sharing schemes. *IEEE Trans. Inf. Theory*, 40(3):786–794, 1994.

[8] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer, 1988.

[9] G. R. Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge, MARK 1979, New York, NY, USA, June 4-7, 1979*, pages 313–318. IEEE, 1979.

[10] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 565–596. Springer, 2018.

[11] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptol.*, 17(4):297–319, 2004.

[12] Allan Borodin and R. Moenck. Fast modular transforms. *J. Comput. Syst. Sci.*, 8(3):366–386, 1974.

[13] Gabriel Bracha. An asynchronous [(n-1)/3]-resilient consensus protocol. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing (PODC), 1984*, pages 154–162, 1984.

[14] Michael R. Capalbo, Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 659–668. ACM, 2002.

[15] 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*, volume 4515 of *Lecture Notes in Computer Science*, pages 291–310. Springer, 2007.

[16] Benny Chor and Eyal Kushilevitz. Secret sharing over infinite domains. *J. Cryptol.*, 6(2):87–95, 1993.

[17] Ronald Cramer, Ivan Bjerre Damgård, Nico Döttling, Serge Fehr, and Gabriele Spini. Linear secret sharing schemes from error correcting codes and universal hash functions. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 313–336. Springer, 2015.

[18] Ronald Cramer and Serge Fehr. Optimal black-box secret sharing over arbitrary abelian groups. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 272–287. Springer, 2002.

[19] Ronald Cramer, Serge Fehr, and Martijn Stam. Black-box secret sharing from primitive sets in algebraic number fields. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 344–360. Springer, 2005.

[20] Ronald Cramer and Chaoping Xing. Blackbox secret sharing revisited: A coding-theoretic approach with application to expansionless near-threshold schemes. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 499–528. Springer, 2020.

[21] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam D. Smith. Scalable multiparty computation with nearly optimal work and resilience. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 241–261, 2008.

[22] Yvo Desmedt. Threshold cryptography. *Eur. Trans. Telecommun.*, 5(4):449–458, 1994.

[23] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, 1989.

[24] Erez Druk and Yuval Ishai. Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In Moni Naor, editor, *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 169–182. ACM, 2014.

[25] Matthias Fitzi, Matthew K. Franklin, Juan A. Garay, and Harsha Vardhan Simhadri. Towards optimal and efficient perfectly secure message transmission. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 311–322, 2007.

[26] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Inf. Theory*, 8(1):21–28, 1962.

[27] Oded Goldreich. A sample of samplers: A computational perspective on sampling. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, volume 6650 of *Lecture Notes in Computer Science*, pages 302–332. Springer, 2011.

[28] Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Trans. Inf. Theory*, 51(10):3393–3400, 2005.

[29] Danny Harnik, Yuval Ishai, and Eyal Kushilevitz. How many oblivious transfers are needed for secure multiparty computation? In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, pages 284–302, 2007.

[30] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 21–30, 2007.

[31] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 433–442. ACM, 2008.

[32] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 572–591, 2008.

[33] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111. IEEE Computer Society, 1993.

[34] Richard Karp, Nicholas Pippenger, and Michael Sipser. A time-randomness tradeoff. In *AMS Conference on Probabilistic Computational Complexity*, volume 111, 1985.

[35] Hugo Krawczyk. Distributed fingerprints and secure information dispersal. In Jim Anderson and Sam Toueg, editors, *Proceedings of the Twelth Annual ACM Symposium on Principles of Distributed Computing, Ithaca, New York, USA, August 15-18, 1993*, pages 207–218. ACM, 1993.

[36] Michael Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, and Daniel A. Spielman. Efficient erasure correcting codes. *IEEE Trans. Inf. Theory*, 47(2):569–584, 2001.

[37] Michael Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, Daniel A. Spielman, and Volker Stemann. Practical loss-resilient codes. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 150–159. ACM, 1997.

[38] James L. Massey. Some applications of source coding in cryptography. *Eur. Trans. Telecommun.*, 5(4):421–430, 1994.

[39] P. Oswald and Amin Shokrollahi. Capacity-achieving sequences for the erasure channel. *IEEE Trans. Inf. Theory*, 48(12):3017–3028, 2002.

[40] Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM Journal on Computing*, 9(2):230–250, 1980.

[41] Thomas J. Richardson and Rüdiger L. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008.

[42] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[43] Victor Shoup. Practical threshold signatures. In *Advances in Cryptology - EUROCRYPT 2000*, pages 207–220. Springer, 2000.

[44] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Found. Trends Theor. Comput. Sci.*, 5(3-4):207–388, 2010.

[45] Alin Tomescu, Robert Chen, Yiming Zheng, Ittai Abraham, Benny Pinkas, Guy Golan-Gueta, and Srinivas Devadas. Towards scalable threshold cryptosystems. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 877–893. IEEE, 2020.