Towards Breaking the Exponential Barrier for General Secret Sharing

Tianren Liu[1⋆], Vinod Vaikuntanathan[1⋆⋆], and Hoeteck Wee[2⋆⋆⋆]

[1] MIT
[2] CNRS and ENS

Abstract. A secret-sharing scheme for a monotone Boolean (access) function $F : \{0,1\}^n \to \{0,1\}$ is a randomized algorithm that on input a secret, outputs n shares $s_1, \ldots, s_n$ such that for any $(x_1, \ldots, x_n) \in \{0,1\}^n$, the collection of shares $\{s_i : x_i = 1\}$ determine the secret if $F(x_1, \ldots, x_n) = 1$ and reveal nothing about the secret otherwise. The best secret sharing schemes for general monotone functions have shares of size $\Theta(2^n)$. It has long been conjectured that one cannot do much better than $2^{\Omega(n)}$ share size, and indeed, such a lower bound is known for the restricted class of linear secret-sharing schemes.

In this work, we *refute* two natural strengthenings of the above conjecture:

- First, we present secret-sharing schemes for a family of $2^{2^{n/2}}$ monotone functions over $\{0,1\}^n$ with sub-exponential share size $2^{O(\sqrt{n}\log n)}$. This *unconditionally* refutes the stronger conjecture that circuit size is, within polynomial factors, a lower bound on the share size.
- Second, we disprove the analogous conjecture for non-monotone functions. Namely, we present "non-monotone secret-sharing schemes" for *every access function* over $\{0,1\}^n$ with shares of size $2^{O(\sqrt{n}\log n)}$.

Our construction draws upon a rich interplay amongst old and new problems in information-theoretic cryptography: from secret-sharing, to multi-party computation, to private information retrieval. Along the way, we also construct the first *multi-party* conditional disclosure of secrets (CDS) protocols for general functions $F : \{0,1\}^n \to \{0,1\}$ with communication complexity $2^{O(\sqrt{n}\log n)}$.

⋆ liutr@mit.edu. Research supported in part by NSF Grants CNS-1350619 and CNS-1414119.

⋆⋆ vinodv@csail.mit.edu. Research supported in part by NSF Grants CNS-1350619 and CNS-1414119, Alfred P. Sloan Research Fellowship, Microsoft Faculty Fellowship, the NEC Corporation, a Steven and Renee Finn Career Development Chair from MIT. This work was also sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236.

⋆⋆⋆ wee@di.ens.fr. Research supported in part by ERC Project aSCEND (H2020 639554) and NSF Award CNS-1445424.

1 Introduction

Secret sharing [Sha79, Bla79] is a powerful cryptographic technique that allows a dealer to distribute shares of a secret to n parties such that certain authorized subsets of parties, and only they, can recover the secret. The original definition of secret sharing is what we now call an (n, t)-threshold secret sharing scheme, where any set of t or more parties can recover the secret, and no subset of fewer than t parties can learn any information about the secret whatsoever.

Later on, this was generalized in [ISN89] to the notion of a secret-sharing scheme realizing a monotone function $F : \{0,1\}^n \to \{0,1\}$. This is simply a randomized algorithm that on input a secret, outputs n shares $s_1, \ldots, s_n$ such that for any $(x_1, \ldots, x_n) \in \{0,1\}^n$, the collection of shares $\{s_i : x_i = 1\}$ determine the secret if $F(x_1, \ldots, x_n) = 1$ and reveal nothing about the secret otherwise.[3] It is easy to see that (n, t)-threshold secret sharing corresponds to the special case where F is the (monotone) threshold function that outputs 1 if and only if at least t of the n input bits are 1.

While the landscape of threshold secret sharing is relatively well-understood, even very basic information-theoretic questions about the more general notion of secret sharing remain embarrassingly open. It is simple to construct a secret sharing scheme realizing any monotone function $F : \{0,1\}^n \to \{0,1\}$ where each share is at most 2^n bits; the share size can be improved to $O(2^n/\sqrt{n})$ bits [ISN89]. We also know that there is an (explicit) monotone function $F : \{0,1\}^n \to \{0,1\}$ that requires a total share size of $\Omega(n^2/\log n)$ bits [Csi97], a far cry from the upper bound. No better lower bounds are known (except for the restricted class of linear secret-sharing schemes, cf. Section 1.3), even in a non-explicit sense.

Closing the exponential gap between the afore-mentioned upper bound or lower bounds is a long-standing open problem in cryptography. The general consensus appears to be that the upper bound is almost tight, as formalized in a conjecture of Beimel [Bei11]:

> **Conjecture (main).** There exists a family of monotone functions $\{F_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ *s.t.* the total share size of any secret sharing scheme realizing F_n is $2^{\Omega(n)}$ bits.

Note that this is a purely information-theoretic statement with no reference to the computational complexity of sharing or reconstruction.

1.1 Our Results

In this work, we *refute* two natural strengthenings of the main conjecture by presenting new secret sharing schemes. The first variant of the main conjecture

[3] The typical formulation of secret-sharing refers to a dealer that holds a secret distributing shares to n parties, such that only certain subsets of parties —described by a so-called access structure— can reconstruct the secret. In our formulation, the randomized algorithm corresponds to the dealer, s_i corresponds to the share given to party i, $x_i \in \{0,1\}$ indicates whether party i is present in a subset, and F corresponds to the access structure.

considers a lower bound on share size that depends on the representation size of the function F as is the case for the state-of-the-art upper bounds, and the second variant considers a natural generalization of secret-sharing to non-monotone functions.

The representation size barrier. To construct a secret-sharing scheme for a function F, we would need some representation of the function F, e.g., as a boolean formula or as a circuit or as a span program [KW93]. The most general approach we have for constructing secret-sharing schemes with small share sizes yields share sizes that are linear in the size of the monotone Boolean formula (more generally, the monotone span program) [BL88]. Recall that there are most $2^{O(s \log s)}$ circuits or formulas or span programs of size s. The following conjecture then captures the intuition that any secret-sharing scheme must yield share sizes that is polynomial in the representation size:

> **Conjecture 1.** For any collection F_n of monotone functions over $\{0,1\}^n$ such that $|F_n| \geq 2^{\omega(n \log n)}$, the total share size of any secret-sharing scheme realizing F_n is at least $(\log |F_n|)^{\Omega(1)}$. [4]

Note that there are $2^{\Theta(2^n/\sqrt{n})}$ monotone functions over $\{0,1\}^n$, so the main conjecture is a special case of Conjecture 1. In addition, by a counting argument, the number of unbounded-fan-in circuits with s gates is no more than $2^{O(s(n+s))}$, thus the collection F_n contains functions whose circuit (and thus formula) size at least $\Omega(\sqrt{\log |F_n|} - n)$. This means that if our intuition that the share size is polynomial in the representation size (as a formula or even a circuit) is correct, then the share size for F_n must be at least $(\log |F_n|)^{\Omega(1)}$, as captured by the conjecture. Indeed, we refute this conjecture.

> **Theorem 1 (informal).** For any $s = s(n) \leq 2^{n/2}$, there is a collection $\hat{F}_{n,s}$ of $2^{s(n)}$ monotone functions over $\{0,1\}^n$ and a secret-sharing scheme for $\hat{F}_{n,s}$ where each share is $2^{O(\sqrt{\log s} \log \log s)} = (\log |\hat{F}_{n,s}|)^{o(1)}$ bits.

In particular, Theorem 1 has the following, we believe surprising, consequences.

First, our result implies that there are secret sharing schemes whose share size is much better than what the "representation size intuition" would suggest. In one extreme case, taking $s(n) = 2^{n/2}$, our result implies a family $\hat{F}_{n,2^{n/2}}$ of $2^{2^{n/2}} = 2^{2^{\Omega(n)}}$ monotone functions and a secret sharing scheme for $\hat{F}_{n,2^{n/2}}$ with share size only $2^{\tilde{O}(\sqrt{n})}$. Whereas, by simple counting arguments, there must be a function in this class with circuits (or formulas or monotone span programs or essentially every other natural computational model we can think of) of size

[4] The same secert-sharing algorithm can be used to realizing as many as $n!$ different access functions by permuting the parties. This trick comes from the nature of secret sharing, thus two access functions is equivalent if one is the composition of a permutation and the other, and Conjecture 1 should be stated on the number of equivalence classes in F_n. Assuming $|F_n| \gg n!$ has essentially the same effect.

$2^{\Omega(n)}$. As another reference point, taking $s(n) = n^{\log n}$, it follows that there exists monotone functions over $\{0,1\}^n$ that require quasi-polynomial (in n) size circuits (and formulas and monotone span programs), but which admit secret-sharing schemes with polynomial share size. This in particular implies that existing secret-sharing schemes with share sizes linear in the formula size are far from optimal.

Second, our result implies that "non-linear reconstruction" totally dominates "linear reconstruction". Secret-sharing schemes with linear reconstruction are known to be equivalent to monotone span programs [KW93], whereas the scheme from Theorem 1 has a *non-linear* reconstruction algorithm. In particular, our results shows that for share size $\mathrm{poly}(n)$ (resp., $2^{\sqrt{n}}$), there are $2^{\mathsf{quasipoly}(n)}$ (resp., $2^{2^{\Omega(n)}}$) access structures that can be realized by secret sharing schemes with non-linear reconstruction, compared to $2^{\mathrm{poly}(n)}$ (resp., $2^{2^{\Omega(\sqrt{n})}}$) by linear schemes.

Prior to this work, such a statement was only known under intractibility assumptions pertaining to number-theoretic and combinatorial problems like quadratic residuosity and graph isomorphism [BI01, VV15], whereas our result is *unconditional*.

Non-monotone secret sharing. A further generalization called *non-monotone* secret sharing was defined in the work of Beimel and Ishai [BI01] and further studied in [BIKK14, VV15]; this is a natural extension of secret sharing to any arbitrary, possibly non-monotone F. A non-monotone secret-sharing scheme for a function $F : \{0,1\}^n \to \{0,1\}$ is a randomized algorithm that on input a secret, outputs n *pairs* of shares $(s_{i,0}, s_{i,1})_{i\in[n]}$ such that for any $(x_1, \ldots, x_n)$, the n shares $(s_{1,x_1}, \ldots, s_{n,x_n})$ determine the secret if $F(x_1, \ldots, x_n) = 1$ and reveal nothing about the secret otherwise. Standard monotone secret-sharing correspond to the special case where F is monotone and $s_{1,0} = \cdots = s_{n,0} = \perp$. Non-monotone secret sharing schemes are natural candidates for use in advanced cryptographic schemes such as attribute-based encryption [GPSW06, OSW07].

It is easy to see that we can construct non-monotone secret-sharing schemes for *all functions* on n bits starting from standard secret-sharing for all monotone functions on $2n$ bits, with a small polynomial blow-up in the share size. This might suggest that the best share sizes for non-monotone secret-sharing and standard secret-sharing are polynomially related, motivating the following strengthening of the main conjecture that we formalize below:

> **Conjecture 2.** There exists a family of functions $\{F_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ such that the total share size in any non-monotone secret sharing scheme for F_n is $2^{\Omega(n)}$ bits.

Indeed, we also refute this conjecture:

> **Theorem 2 (informal).** There is a non-monotone secret-sharing for the family of all functions $F : \{0,1\}^n \to \{0,1\}$ where each share is $2^{\tilde{O}(\sqrt{n})}$ bits.

1.2 Overview of Our Constructions

We derive both Theorems 1 and 2 from the construction of a more general cryptographic primitive, namely that of conditional disclosure of secrets (CDS) [GIKM00], which is a generalization of non-monotone secret-sharing to general, non-boolean inputs. Informally, conditional disclosure of secrets allows a set of parties to disclose a secret to an external party Charlie, subject to a given condition on their joint inputs. Concretely, we consider $(k+1)$-party CDS for $\mathbf{INDEX}_N$, where Alice holds $\mathbf{D} \in \{0,1\}^N$, parties $P_1, \ldots, P_k$ "jointly" hold an index $i \in [N]$[5], and all of them hold a secret $\mu \in \{0,1\}$, and Charlie knows $\mathbf{D}, i$ and should learn μ iff $\mathbf{D}[i] = 1$. To enable this, Alice and all the parties should share randomness that is hidden from Charlie (akin to the random coins used in a secret-sharing scheme). Our goal is to minimize the communication complexity in CDS, that is, the total number of bits sent by Alice and the k parties to Charlie.

Our main result is as follows:

> **Theorem (main).** For any $1 \leq k \leq \log N$, there is a $(k+1)$-party CDS for $\mathbf{INDEX}_N$ where the total communication complexity is $2^{O(\sqrt{\log N} \cdot \log\log N)}$ bits.

Previously, such a result was only known for $k = 1$ [LVW17]. Before describing our $(k+1)$-party CDS, we briefly explain how Theorems 1 and 2 follow from the CDS.

Our non-monotone secret-sharing scheme for all functions $F : \{0,1\}^n \to \{0,1\}$ in Theorem 2 follows from the special case $k = \log N$, where $N = 2^n$. Concretely, the non-monotone secret-sharing scheme for F is derived from the $(n+1)$-party CDS for $\mathbf{INDEX}_{2^n}$ as follows: Alice holds the truth table $\mathbf{D} \in \{0,1\}^{2^n}$ of a function $F : \{0,1\}^n \to \{0,1\}$ and each party $P_i, i = 1, \ldots, n$ holds a single bit of the index $i \in [2^n]$, and the messages sent by P_i in the CDS corresponds to the shares. Going from the $(n+1)$-party CDS to Theorem 1 requires an additional folklore transformation which transforms a *non-monotone* secret-sharing scheme for any $F : \{0,1\}^n \to \{0,1\}$ into a *monotone* secret-sharing scheme with roughly the same share size for a related function F'; see Section 6.3 for details.

A general framework for CDS. We proceed to provide an overview for our $(k+1)$-party CDS protocol. We begin with a general framework for constructing CDS protocols, and then sketch how to instantiate the underlying building blocks to obtain our main result.

The LVW17 framework. We begin by sketching the 2-party CDS protocol, i.e. $k = 1$, from [LVW17] (which in turn builds upon [BIKK14, DG15]). The

[5] We will make the precise sense of how the parties "jointly" hold the index clear in a little bit, but roughly speaking, the reader should imagine that each party holds $\lceil (\log N)/k \rceil$ bits of the index.

starting point of their protocol is a notion of (N, ℓ)-PIR encoding, which encodes $i \in [N]$ as vector $\mathbf{u}_i \in \mathbb{Z}_6^\ell$ and $\mathbf{D}$ as a function $H_{\mathbf{D}} : \mathbb{Z}_6^\ell \to \mathbb{Z}_6^\ell$ such that for all $i, \mathbf{D}, \mathbf{w}$, we have

$$\mathbf{D}[i] = \langle H_{\mathbf{D}}(\mathbf{u}_i + \mathbf{w}), \mathbf{u}_i \rangle - \langle H_{\mathbf{D}}(\mathbf{w}), \mathbf{u}_i \rangle.$$

This immediately implies that for all $\mu \in \{0, 1\}$, we have

$$\mu\mathbf{D}[i] = \langle H_{\mathbf{D}}(\mu\mathbf{u}_i + \mathbf{w}), \mathbf{u}_i \rangle - \langle H_{\mathbf{D}}(\mathbf{w}), \mathbf{u}_i \rangle. \tag{1}$$

[LVW17] constructed a two-party CDS protocol with communication $O(\ell)$ starting from any (N, ℓ)-PIR encoding, and also gave a construction of a $(N, 2^{\tilde{O}(\sqrt{\log N})})$-PIR encoding. The two-party CDS protocol is as follows:

- Alice and P_1 share randomness $\mathbf{w}, \mathbf{r}$ (hidden from Charlie);
- Alice sends $\mathbf{m}_A^1 := H_{\mathbf{D}}(\mathbf{w}) + \mathbf{r}$.
- P_1 sends $\mathbf{m}_B^1 := \mu\mathbf{u}_i + \mathbf{w}$ and $m_B^2 := \langle \mathbf{u}_i, \mathbf{r} \rangle$.
- Charlie can now compute $\mu\mathbf{D}[i]$ (and thus μ) given $\mathbf{D}, i, (\mathbf{m}_A^1, \mathbf{m}_B^1, m_B^2)$ using the relation

$$\mu\mathbf{D}[i] = \langle H_{\mathbf{D}}(\underbrace{\mu\mathbf{u}_i + \mathbf{w}}_{\mathbf{m}_B^1}), \mathbf{u}_i \rangle - \langle \underbrace{H_{\mathbf{D}}(\mathbf{w}) + \mathbf{r}}_{\mathbf{m}_A^1}, \mathbf{u}_i \rangle + \underbrace{\langle \mathbf{r}, \mathbf{u}_i \rangle}_{m_B^2}$$

  which follows readily from (1).

It is easy to see that the total communication is $O(\ell)$. Privacy follows fairly readily from the fact that the joint distribution of $(\mathbf{m}_A^1, \mathbf{m}_B^1)$ is uniformly random, and that m_B^2 is completely determined given $(\mathbf{m}_A^1, \mathbf{m}_B^1)$ and $\mu\mathbf{D}[i]$ along with $\mathbf{D}, i$.

The multi-party setting. We show how to extend the [LVW17] construction to the multi-party setting with k parties, for any $k \geq 1$. Here, the index i is distributed across k parties $P_1, \ldots, P_k$. The key idea is to have these k parties jointly emulate P_1 in the two-party CDS via secure multi-party computation (MPC); in fact, because of communication constraints, we will use a *private simultaneous messages* protocol [FKN94, IK97] in this setting, where each of the k parties sends a single message to Charlie. That is, these k parties jointly hold inputs $i, \mathbf{w}, \mathbf{r}, \mu$ and they will run an MPC protocol with Charlie so that Charlie learns

$$(\mu\mathbf{u}_i + \mathbf{w}, \langle \mathbf{u}_i, \mathbf{r} \rangle), \tag{2}$$

upon which Charlie can proceed as in the two-party CDS to recover $\mu\mathbf{D}[i]$. Moreover, security of the MPC protocol ensures that what Charlie learns in the k-party CDS is the same as that in the two-party CDS. Correctness and security then follow readily from those of the two-party CDS and the MPC protocol.

Recall that our goal is to obtain a protocol with total communication complexity $2^{o(n)}$, and we need to make sure that the MPC protocol does not blow

up the communication by too much. The key insight is that the total size of the inputs for the MPC protocol is $O(\log N + \ell)$ and is in particular independent of $\mathbf{D}$. Therefore, it suffices to design an MPC protocol for computing (2) with polynomial communication for the $(N, 2^{\tilde{O}(\sqrt{\log N})})$-PIR encoding in [LVW17], upon which we will derive a k-party CDS protocol with total communication $\mathrm{poly}(\ell) = 2^{\tilde{O}(\sqrt{\log N})}$.

Minimizing communication via decomposability. Prior works on MPC tells us that the communication cost for securely computing (2) is essentially dominated by the cost of (non-securely) computing the k-ary functionality

$$i = (i_1, \ldots, i_k) \mapsto \mathbf{u}_i.$$

In fact, it suffices to construct PIR-encodings where $\mathbf{u}_i \in \mathbb{Z}_6^\ell$ may be derived by applying a simple function to vectors $\mathbf{u}_{i_1}, \ldots, \mathbf{u}_{i_k} \in \mathbb{Z}_6^\ell$, each of which is derived from some local (and possibly complex) computation on $i_1, \ldots, i_k$ respectively. In this work, we consider $\mathbf{u}_i$ that are given by

$$\mathbf{u}_i = \mathbf{u}_{1,i_1} \circ \cdots \circ \mathbf{u}_{k,i_k}$$

where $\circ$ corresponds to point-wise product of vectors. We refer to this property as *k-decomposability*.

Using k-decomposable $\mathbf{u}_i$, the computation in (2) can be written as

$$(i_1, \ldots, i_k, \mathbf{w}, \mathbf{r}, \mu) \mapsto (\mu \mathbf{u}_{1,i_1} \circ \cdots \circ \mathbf{u}_{k,i_k} + \mathbf{w}, \langle \mathbf{u}_{1,i_1} \circ \cdots \circ \mathbf{u}_{k,i_k}, \mathbf{r} \rangle)$$

which is essentially a degree $k+1$ computation over the inputs; concretely, it can be written as the sum of a small number of monomials over $(\mathbf{u}_{i_1}, \ldots, \mathbf{u}_{i_k}, \mathbf{w}, \mathbf{r}, \mu)$. Following [IK00, IK02, CFIK03], such a computation admits a non-interactive MPC protocol satisfying perfect, information-theoretic security, and total communication polynomial in $\ell, k, \log N$.

This brings us to the final building block: a $(N, 2^{\tilde{O}(\sqrt{\log N})})$-PIR encoding that is k-decomposable.

PIR-Encodings from Matching Vector (MV) Families. The key tool in the $(N, 2^{\tilde{O}(\sqrt{\log N})})$-PIR encoding in [LVW17] is matching vector (MV) families, first constructed by Grolmusz [Gro00] and introduced to cryptography in the context of private information retrieval [Yek08, Efr12, DGY11, DG15].

MV families. A (mod 6) MV family is an explicit collection of vectors $\{(\mathbf{u}_i, \mathbf{v}_i)\}_{i \in [N]}$ such that $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{Z}_6^\ell$ where $\ell = 2^{O(\sqrt{\log N} \cdot \log\log N)}$ and:

$$\begin{aligned} &\langle \mathbf{u}_i, \mathbf{v}_i \rangle = 0, \\ &\langle \mathbf{u}_i, \mathbf{v}_j \rangle \in \{1, 3, 4\} \quad \text{for } i \neq j. \end{aligned}$$

where all computations are done mod 6.

At this point, it may seem like we are abusing notation as we are using $\mathbf{u}_i$ to denote both the vectors in a MV family and those in a PIR encoding. Fortunately, they are essentially the same thing in the [LVW17] construction, and therefore, it suffices to construct MV families where the underlying $\mathbf{u}_i$'s are k-decomposable.

Prior constructions. We begin with an overview of Grolmusz's MV families [Gro00]. Fix any integers h, w so that $\binom{h}{w} \geq N$. Pick *any* distinct $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^h$ of Hamming weight exactly w. Observe that for all $i, j \in [N]$:

$$\|\mathbf{x}_i \circ \mathbf{x}_j\|_1 \begin{cases} = w & \text{if } i = j \\ < w & \text{if } i \neq j \end{cases}$$

The vectors $\mathbf{u}_i$ will have length $\ell = h^{O(\sqrt{w})}$ indexed by subsets S of $[h]$ of size at most $O(\sqrt{w})$ and defined as follows:

$$\mathbf{u}_i[S] = \prod_{i' \in S} \mathbf{x}_i[i']$$

The reason for defining $\mathbf{u}_i$ this way is that for any fixed polynomial f of degree $O(\sqrt{w})$ from $\{0,1\}^h \to \mathbb{Z}_6$, we can write $f(\mathbf{x}_i \circ \mathbf{x}_j)$ as $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$ where $\mathbf{v}_j$ depends only on f and $\mathbf{x}_j$. (Roughly speaking, the polynomial f checks whether the Hamming weight of its input is equal to or less than w.) We can then set $h = 2 \log N, w = \log N$, which yields $\ell = 2^{O(\sqrt{\log N} \cdot \log \log N)}$.

Our construction. In our setting, the index $i = (i_1, \ldots, i_k)$ is divided equally amongst k parties and want $\mathbf{u}_i$ to be of the form

$$\mathbf{u}_i = \mathbf{u}_{1,i_1} \circ \cdots \circ \mathbf{u}_{k,i_k}.$$

To achieve this property, it suffices to modify the choices of $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in the prior construction. Concretely, we pick w, h to be multiples of k such that $\binom{h/k}{w/k}^k \geq N$. Then, we choose $\mathbf{x}_1, \ldots, \mathbf{x}_N$ so that each $\mathbf{x}_i$ can be decomposed into k blocks $(\mathbf{x}_{i_1} \| \cdots \| \mathbf{x}_{i_k})$ each of weight exactly w/k. Recall that each entry of $\mathbf{u}_i$ is a product of $O(\sqrt{w})$ entries of $\mathbf{x}_i$, which means we can write $\mathbf{u}_i = \mathbf{u}_{1,i_1} \circ \cdots \circ \mathbf{u}_{k,i_k}$ where each $\mathbf{u}_{1,i_1}, \ldots, \mathbf{u}_{k,i_k}$ depends on $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}$ respectively. We can still set $h = 2 \log N, w = \log N$ as before, since $\binom{h/k}{w/k}^k \geq ((\frac{h}{w})^{w/k})^k = N$ for any $1 \leq k \leq \log N$.

1.3 Related Work

A linear secret-sharing scheme is one where the secret sharing algorithm computes a linear function of the secret and its randomness. Most secret-sharing schemes in the literature are linear secret-sharing schemes, and many cryptographic applications also require the linearity property. For linear secret-sharing

schemes, the existing upper bounds (namely, linear in formula or span program size) are essentially optimal, due to their connection to a computational model called monotone span programs defined by Karchmer and Wigderson [KW93].

Using this connection, we know the following results about linear secret-sharing schemes. We know there exist access functions that need $2^{\Omega(n)}$ share size for linear secret sharing via a counting argument [KW93, Bei11]. As for explicit functions, we have $n^{\log n/\log\log n}$ lower bounds for linear secret sharing realizing some explicit access functions [BGP95, BGW99]. Quite recently, this was improved by Pitassi and Robere [RPRC16, PR17] who showed an *exponential* lower bound for monotone span programs realizing some explicit access functions. Therefore, to beat these exponential bounds, as we do in this work, we need to turn to general, *non-linear* secret-sharing schemes.

1.4 Discussion

Our work highlights new connections and exploits the rich interplay amongst old and new problems in information-theoretic cryptography: from secret-sharing (70s), to multi-party computation (80s), to private information retrieval (90s), and brings forth strong evidence against the conjectured optimality of the classic constructions for monotone secret-sharing.

While we do not construct secret-sharing schemes with sub-exponential share sizes for all monotone functions over $\{0,1\}^n$, as would be necessary to refute the main conjecture, we do achieve sub-exponential $2^{O(\sqrt{n}\log n)}$ share sizes for a large number of these functions, namely $2^{2^{n/2}}$ out of $2^{2^{n-O(\log n)}}$ of them. There are several very exciting new research directions at this point, and we highlight two specific questions related to the main conjecture:

- Does there exist a family containing 1% of all monotone functions over $\{0,1\}^n$ that admit a monotone secret-sharing scheme of total share size $2^{o(n)}$? Here, 1% can be replaced by any constant. One way to resolve this question would be to extend our construction to a *larger* set of monotone functions.
- Does there exist a family of $2^{2^{\Omega(n)}}$ functions over $\{0,1\}^n$ that admit a monotone secret-sharing scheme of total share size $2^{o(\sqrt{n})}$? One way to resolve this question would be to *improve* the communication complexity of 2-party CDS, which in turn seems closely related to the problem of improving the communication complexity of the state-of-the-art 2-server private information retrieval.

On another thread, we note that the work of Beimel, Ishai, Kumaresan and Kushilevitz [BIKK14] showed ways to use improved PIR schemes to obtain protocols for various information-theoretic multiparty tasks improving their communication complexity or randomness complexity. Our work continues this line of thought. We mention that the problem of improving the communication complexity of private simultaneous messages (PSM) protocols, a a generalization of CDS, for general functions to a sub-exponential number remains wide open.

1.5 Organization

We start with Section 3 which describes the framework of the multiparty CDS construction. That is, a multi-party CDS scheme for the **INDEX**predicate can be constructed from: a) a "PIR-encoding", and b) a private simultaneous messages (PSM) protocol computing a special functionality related to the PIR-encoding (see Theorem 3.1). The following two sections construct these two building blocks. Section 4 shows there exists a succinct PIR-encoding that is also decomposable (Theorem 4.9). Section 5 shows there exists an efficient PSM for the special functionality if the PIR-encoding is decomposable (Theorem 5.1). An immediate combination of the above results gives an good multi-party CDS protocol (Theorem 6.1, which is a restatement of the main theorem in the introduction). Section 6 also presents our applications to secret sharing. A good multi-party CDS protocol implies non-monotone secret sharing for all functions (Theorem 6.5, which is a restatement of Theorem 2 in the introduction). It also implies monotone secret sharing for a large class of functions (Theorem 6.7, which is a restatement of Theorem 1 in the introduction). In addition, Section 7 shows similar results for linear CDS and linear secret sharing.

2 Preliminaries and Definitions

We start with some notation that we will use throughout the paper.

- Let $\mathcal{R}$ denote a generic commutative ring, $\mathbb{Z}$ denote the integer ring, and $\mathbb{Z}_m$ denote the ring of integers modulo m. Let $\mathbb{F}$ denote a generic finite field. When q is a prime power, let $\mathbb{F}_q$ denote the finite field of size q. For an integer m, let $[m] := \{1, \ldots, m\}$.
- We will let boldface letters, such as $\mathbf{x}$, denote vectors. When $\mathbf{x}$ is a vector, let $\mathbf{x}[i]$ denote its i-th element.
- For vectors $\mathbf{x} \in \mathcal{R}^\ell, \mathbf{y} \in \mathcal{R}^{\ell'}$, let $\mathbf{x} \| \mathbf{y} \in \mathcal{R}^{\ell+\ell'}$ denote their concatenation.
- Call a vector $\mathbf{x} \in \mathcal{R}^\ell$ a *zero-one vector* if its entries are either 0 or 1. For a zero-one vector $\mathbf{x}$, let $\|\mathbf{x}\|_1$ denote the number of 1's in $\mathbf{x}$.

Definition 2.1 (point-wise product). *For any two vectors $\mathbf{x}, \mathbf{y} \in \mathcal{R}^\ell$, their point-wise product, denoted by $\mathbf{x} \circ \mathbf{y}$, is a vector in the same linear space whose i-th element is the product of the i-th elements of $\mathbf{x}, \mathbf{y}$, i.e. $(\mathbf{x} \circ \mathbf{y})[i] = \mathbf{x}[i] \cdot \mathbf{y}[i]$.*

This is also known in the literature as the Hadamard product or Schur product, typically used in the context of matrices.

2.1 k-party Conditional Disclosure of Secrets (CDS)

In a k-party CDS scheme, there are k parties who know a secret message μ and jointly hold input x. These parties cannot communicate with each other, but instead they have access to a common random string (CRS). Their goal is to send a single message to the CDS referee Charlie, at the end of which Charlie, who already knows x, should learn μ if and only if $\mathsf{P}(x) = 1$, for a fixed predicate P.

Definition 2.2 (conditional disclosure of secrets (CDS) [GIKM00]). *Let input spaces $\mathcal{X}_1, \ldots, \mathcal{X}_k$, secret space $\mathcal{M}$ and randomness space $\mathcal{W}$ be finite sets. Fix a predicate $\mathsf{P} : \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_k \to \{0,1\}$. A* cc-conditional disclosure of secrets (CDS) *protocol for P is a tuple of deterministic functions $(\mathsf{B}_1, \ldots, \mathsf{B}_k, \mathsf{C})$*

$$\textit{Transmitting functions } \mathsf{B}_i : \mathcal{M} \times \mathcal{X}_i \times \mathcal{W} \to \{0,1\}^{\mathsf{cc}}$$
$$\textit{Reconstruction function } \mathsf{C} : \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \times \{0,1\}^{\mathsf{cc} \times k} \to \mathcal{M}$$

satisfying the following properties:

(reconstruction.) *For all $(x_1, \ldots, x_k) \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_k$ such that $\mathsf{P}(x_1, \ldots, x_k) = 1$, for all $w \in \mathcal{W}$, and for all $\mu \in \mathcal{M}$:*

$$\mathsf{C}(x_1, \ldots, x_k, \mathsf{B}_1(\mu, x_1; w), \ldots, \mathsf{B}_k(\mu, x_k; w)) = \mu \ .$$

(privacy.) *There exists a randomized algorithm S such that for all input tuple $(x_1, \ldots, x_k) \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_k$ satisfying $\mathsf{P}(x_1, \ldots, x_k) = 0$, the joint distribution of $\big(\mathsf{B}_1(\mu, x_1; w), \ldots, \mathsf{B}_k(\mu, x_k; w)\big)$ is perfectly indistinguishable from $\mathsf{S}(x_1, \ldots, x_k)$, where the randomness are taken over $w \xleftarrow{R} \mathcal{W}$ and the coin tosses of S.*

Predicates. We consider the following predicates:

- Index $\mathbf{INDEX}_N^{k+1}$: An index $i \in [N]$ is distributed amongst the first k parties. Let $\mathcal{X}_1 = \cdots = \mathcal{X}_k := [\sqrt[k]{N}]$, $\mathcal{X}_{k+1} = \{0,1\}^N$ and under the natural mapping $[N] \ni i \mapsto (i_1, \ldots, i_k) \in ([\sqrt[k]{N}])^k$,

  $$\mathsf{P}_{\mathbf{INDEX}}(i_1, \ldots, i_k, \mathbf{D}) = 1 \text{ iff } \mathbf{D}[i] = 1$$

  Note that $\mathbf{D}$ can also be interpreted as the characteristic vector of a subset of $[N]$.
- All ("worst") predicates $\mathbf{ALL}_N^k$: An index $i \in [N]$ is distributed among the k parties as before and the predicate is specified by a truth table. Let $\mathcal{X}_1 = \ldots = \mathcal{X}_k := [\sqrt[k]{N}]$ and there is a fixed public function $F : [N] \to \{0,1\}$. under the natural mapping $i \in [N] \mapsto (i_1, \ldots, i_k) \in [\sqrt[k]{N}]^k$,

  $$\mathsf{P}_{\mathbf{ALL}}(i_1, \ldots, i_k) := F(i).$$

  $\mathbf{ALL}_N^k$ is an easier predicate (family) than $\mathbf{INDEX}_N^{k+1}$, as any CDS protocol for $\mathbf{INDEX}_N^{k+1}$ implies a CDS protocol for $\mathbf{ALL}_N^k$ with the same total communication complexity (e.g. [LVW17, Section 2.3]).

The definitions of both predicates inherently require N to be a perfect k-th power. In the case where N is not a k-th power, we can pad N to the nearest larger k-th power. When $k \leq \log N$, it's guaranteed that the nearest larger k-th power is no greater than N^2. For the sake of our result, a square blowup on N doesn't matter.

2.2 k-party Private Simultaneous Messages (PSM)

In a k-party PSM scheme, there are k parties who jointly have input x, and they cannot communicate with each other, but have access to a common random string (CRS), as in the case of CDS. There is also the PSM referee Charlie who wants to learn $\mathsf{F}(x)$, for a fixed functionality F. In the PSM scheme, every party sends a single message to Charlie based on its piece of input and the CRS. Given these messages, Charlie should be able to learn $\mathsf{F}(x)$ and nothing else about x.

Definition 2.3 (private simultaneous message (PSM)). *Let $\mathcal{X}_t$ be the input space of the t-th party, let $\mathcal{X} \subseteq \mathcal{X}_1 \times \ldots \times \mathcal{X}_k$ be the input space, and let $\mathcal{M}$ be the output space. Fix a functionality $\mathsf{F} : \mathcal{X} \to \mathcal{M}$. A cc-bits* private simultaneous message (PSM) *protocol for F is a tuple of deterministic functions $(\mathsf{B}_1, \ldots, \mathsf{B}_k, \mathsf{C})$:*

$$\begin{aligned} &\textit{Transmitting functions } \mathsf{B}_i : \mathcal{X}_i \times \mathcal{W} \to \{0,1\}^{\mathsf{cc}}, \\ &\textit{Reconstruction function } \mathsf{C} : \{0,1\}^{\mathsf{cc} \times k} \to \mathcal{M} \end{aligned}$$

satisfying the following properties:

(reconstruction.) *For all $(x_1, \ldots, x_k) \in \mathcal{X}$:*

$$\mathsf{C}(\mathsf{B}_1(x_1; w), \ldots, \mathsf{B}_k(x_k; w)) = \mathsf{F}(x_1, \ldots, x_k)$$

(privacy.) *There exists a randomized simulator S, such that for any input tuple $(x_1, \ldots, x_k) \in \mathcal{X}$, the joint distribution $(\mathsf{B}_1(x_1; w), \ldots, \mathsf{B}_k(x_k; w))$ is perfectly indistinguishable from $\mathsf{S}(\mathsf{F}(x_1, \ldots, x_k))$, where the distributions are taken over $w \xleftarrow{R} \mathcal{W}$ and the coin tosses of S.*

Common Input. In the PSM functionalities we care about in this work, a part of the input is shared among all parties. That is, the input of the t-th party is of the form $x'_t = (x_t, y)$, where x_t is t-th party's exclusive input, and y is shared input known by all parties but the referee Charlie. This can be formalized by letting $\mathcal{X}'_t = \mathcal{X}_t \times \mathcal{Y}$ as the t-th party's input space and by defining the global input space $\mathcal{X}$ as consisting of vectors $((x_1, y_1), \ldots, (x_k, y_k))$ where $y_1 = \ldots = y_k$. For notational simplicity, let $\mathsf{F}(x_1, \ldots, x_k; y)$ denotes $\mathsf{F}((x_1, y), \ldots, (x_k, y))$ and let the transmission functions be denoted as $\mathsf{B}_i(x_i; y; w)$.

Functionality. We consider the following functionalities of interest:

- Affine functions $\mathbf{AFFINE}^k$: For vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathcal{R}^n$ and any affine function $f : \mathcal{R}^{kn} \to \mathcal{R}$

$$\mathsf{F}_{\mathbf{AFFINE}}(\mathbf{x}_1, \ldots, \mathbf{x}_k; f) = f(\mathbf{x}_1, \ldots, \mathbf{x}_k).$$

- Branching Program $\mathbf{BP}^k_m$: A mod-$\mathcal{R}$ branching program is a directed acyclic graph with a source s and a sink t, where every edge is labeled with an affine function. Given an input vector $\mathbf{x}$, the value of an edge is the value of its

label function when applied to $\mathbf{x}$; the value of an (s,t)-path is the product of the values on its edges; and the value of the branching program is the sum of the values of all (s,t)-paths.
To formalize the branching program problem as a PSM functionality: let the branching program be the shared input; split input vector $\mathbf{x}$ among the parties as their exclusive input. More precisely, in $\mathbf{BP}^k_m$, there are k parties and a branching program with m nodes. Let $f_{i,j}$ $(1 \le i < j \le m)$ denote the affine function assigned to edge (i,j). The t-th party's input is $\mathbf{x}_t \in \mathcal{R}^n$ and $\{f_{i,j}\}_{i<j}$. The functionality $\mathbf{BP}^k_m$ is defined as

$$\mathsf{F}_{\mathbf{BP}}(\mathbf{x}_1,\ldots,\mathbf{x}_k;\{f_{i,j}\}_{i<j}) = \sum_{s\text{-}t \text{ path } p} \prod_{\substack{\text{edge } (i,j)\\ \text{in } p}} f_{i,j}(\mathbf{x}_1,\ldots,\mathbf{x}_k).$$

The affine function functionality $\mathbf{AFFINE}^k$ is the special case of $\mathbf{BP}^k$ when the associated graph has only two nodes and one edge.

3 A Framework for Multi-Party CDS

In this section, we describe a framework for constructing multi-party conditional disclosure of information (CDS) protocols. Our framework relies on vector families that satisfy two properties described below. The first is the property of being a "PIR encoding", satisfied by matching vector families and was used in [LVW17] to construct two-party CDS. The second is the existence of a communication-efficient private simultaneous messages (PSM) protocol for a functionality associated to the PIR encoding scheme.

PIR encoding. We define an (N,ℓ)-PIR encoding as a family of N vectors $(\mathbf{u}_i)_{i\in[N]}$ where each $\mathbf{u}_i \in \mathbb{Z}_6^\ell$, along with a mapping $H_{\mathbf{D}} : \mathbb{Z}_6^\ell \to \mathbb{Z}_6^\ell$ for every table (string) $\mathbf{D} \in \{0,1\}^N$. The PIR encoding property requires that for any index $i \in [N]$ and any vector $\mathbf{w} \in \mathbb{Z}_6^\ell$,

$$\langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{w}+\mathbf{u}_i) - H_{\mathbf{D}}(\mathbf{w})\rangle = \mathbf{D}[i]\cdot\phi_{\mathbf{w},i} \tag{3}$$

where $\phi_{\mathbf{w},i} \neq 0$.

As the name suggests, any PIR encoding scheme can be used to construct a 2-server information theoretic private information retrieval (PIR) scheme. Indeed, the 2-server PIR schemes of [CKGS98, WY05, DG15] can all be viewed as instances of this paradigm.

A Special-Purpose PSM Protocol. A PIR encoding scheme can also be used to construct the following CDS protocol as described in [LVW17]. Alice holds a database $\mathbf{D}$, and Bob holds an index i and a secret message μ. Bob sends the pair $(\mathbf{w}+\mu\mathbf{u}_i, \langle \mathbf{u}_i, \mathbf{r}\rangle)$, and Alice sends $H_{\mathbf{D}}(\mathbf{w})+\mathbf{r}$ to Charlie. Here, $\mathbf{w}$ and $\mathbf{r}$ come from the CRS. Charlie now has enough information to compute

$$\langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{w}+\mu\mathbf{u}_i) - H_{\mathbf{D}}(\mathbf{w})\rangle = \mathbf{D}[i]\cdot\mu\cdot\phi_{\mathbf{w},i}$$

which reveals the secret μ if and only if $\mathbf{D}[i] \neq 0$.

In $(k+1)$-party CDS for $\mathbf{INDEX}_N$, the index $i = (i_1, \ldots, i_k)$ is divided equally among the first k parties and the $(k+1)^{th}$ party holds the database $\mathbf{D}$. Our plan is to have the first k parties simulate what Bob did in the 2-party CDS. We describe our CDS protocol in Figure 1, which assumes a PSM protocol for computing the k-party functionality

$$\begin{aligned} &F_{\mathsf{aux}} : [\sqrt[k]{N}] \times \ldots \times [\sqrt[k]{N}] \times (\{0,1\} \times \mathbb{Z}_6^\ell \times \mathbb{Z}_6^\ell) \to \mathbb{Z}_6^\ell \times \mathbb{Z}_6 \\ &\text{where } F_{\mathsf{aux}}(i_1, \ldots, i_k; (\mu, \mathbf{w}, \mathbf{r})) \mapsto (\mathbf{w} + \mu \mathbf{u}_i, \langle \mathbf{u}_i, \mathbf{r} \rangle) \end{aligned} \tag{4}$$

Here, $\mathbf{w}, \mathbf{r}$ and μ are common inputs and the index i is divided equally among the first k parties. This, in particular, will enable the k parties to simulate the Bob in the 2-party CDS setting, and jointly send $\mathbf{w} + \mu \mathbf{u}_i$ and $\langle \mathbf{u}_i, \mathbf{r} \rangle$ to Charlie without revealing any extra information. More precisely, our construction requires a PIR encoding such that there is a PSM for this functionality with communication complexity $\mathsf{cc}_{\mathsf{PSM}}(N, \ell, k)$ that is as small as possible.

Theorem 3.1. *Assume that there is an (N, ℓ)-PIR encoding scheme $(\mathbf{u}_i)_{i \in [N]}$ and a PSM for F_{aux} in (4) with communication complexity $\mathsf{cc}_{\mathsf{PSM}}(N, \ell, k)$, then there is a $(k+1)$-party CDS protocol for $\mathbf{INDEX}_N^{k+1}$ (Figure 1) with communication complexity $\ell + \mathsf{cc}_{\mathsf{PSM}}(N, \ell, k)$.*

CDS for $\mathbf{INDEX}_N^{k+1}$

Building blocks: An (N, ℓ)-PIR encoding scheme $(\mathbf{u}_i)_{i=1}^N$ and a PSM protocol $(\mathsf{B}_1, \ldots, \mathsf{B}_{k+1}, \mathsf{C})$ for F_{aux} in (4).
Input of P_t $(1 \leq t \leq k)$: $i_t \in [N^{1/k}]$
Input of P_{k+1} (Alice): $\mathbf{D} \in \{0,1\}^N$.
Shared Randomness: $\mathbf{w}, \mathbf{r} \in \mathbb{Z}_6^\ell$, $\mu \in \{0,1\}$ and randomness s for the PSM.

The protocol proceeds as follows.

- For $1 \leq t \leq k$, the t-th party sends $\mathbf{m}_{\mathrm{psm},t} := \mathsf{B}_t(i_t; (\mu, \mathbf{w}, \mathbf{r}); s)$.
- P_{k+1} (Alice) sends $\mathbf{m}_A := \mathbf{r} + H_{\mathbf{D}}(\mathbf{w})$.
- Charlie computes $(\mathbf{m}_B^1, m_B^2) := \mathsf{C}(\mathbf{m}_{\mathrm{psm},1}, \ldots, \mathbf{m}_{\mathrm{psm},k})$.
- Charlie outputs 1 if $\langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{m}_B^1) \rangle - \langle \mathbf{u}_i, \mathbf{m}_A \rangle + m_B^2 \neq 0$, and outputs 0 otherwise.

Fig. 1. $(k+1)$-party CDS for $\mathbf{INDEX}_N^{k+1}$ from PIR encodings and PSM.

Proof. By the definition of PIR encoding, for any table $\mathbf{D} \in \{0,1\}^N$, there exists a mapping $H_{\mathbf{D}} : \mathbb{Z}_6^\ell \to \mathbb{Z}_6^\ell$ that satisfies equation (3). We now show correctness, privacy and efficiency of the protocol.

Correctness. The correctness of the PSM protocol tells us that $\mathbf{m}_B^1 = \mathbf{w} + \mu\mathbf{u}_i$ and $m_B^2 = \langle \mathbf{u}_i, \mathbf{r}\rangle$. Equation (3) then directly implies that

$$\langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{w} + \mu\mathbf{u}_i)\rangle - \langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{w})\rangle = \mathbf{D}[i] \cdot \mu \cdot \phi_{\mathbf{w},i}$$

where $\mu \in \{0, 1\}$ is the secret message. Charlie learns

$$\begin{aligned}
&\langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{m}_B^1)\rangle - \langle \mathbf{u}_i, \mathbf{m}_A\rangle + m_B^2 \\
&\quad = \langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{w} + \mu\mathbf{u}_i)\rangle - \langle \mathbf{u}_i, \mathbf{r} + H_{\mathbf{D}}(\mathbf{w})\rangle + \langle \mathbf{u}_i, \mathbf{r}\rangle \\
&\quad = \langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{w} + \mu\mathbf{u}_i)\rangle - \langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{w})\rangle \\
&\quad = \mathbf{D}[i] \cdot \mu\phi_{\mathbf{w},i},
\end{aligned}$$

which, since $\phi_{\mathbf{w},i} \neq 0$, gives μ if and only if $\mathbf{D}[i] = 1$.

Privacy. Privacy follows by putting the following observations together.

- First, the joint distribution of $\mathbf{m}_B^1$ and $\mathbf{m}_A$ is uniformly random, since we are using $(\mathbf{w}, \mathbf{r})$ as one-time pads;
- Secondly, when $\mathbf{D}[i] = 0$, we have $\langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{w} + \mu\mathbf{u}_i)\rangle - \langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{w})\rangle = 0$. This means that $m_B^2 = \langle \mathbf{u}_i, \mathbf{m}_A\rangle - \langle \mathbf{u}_i, H_{\mathbf{D}}(\mathbf{m}_B^1)\rangle$ and thus can be simulated knowing only $\mathbf{m}_A$ and $\mathbf{m}_B^1$ (and, of course, $\mathbf{D}$ and i);
- Finally, the joint distribution of $\mathbf{m}_{\mathrm{psm},1}, \ldots, \mathbf{m}_{\mathrm{psm},k}$ can be perfectly simulated from $(\mathbf{m}_B^1, m_B^2)$, due to the privacy of the PSM.

Efficiency. Each party except Alice sends a PSM message of size at most $\mathsf{cc}_{\mathsf{PSM}}(N, \ell, k)$. Alice sends a vector of size ℓ. The communication complexity of a party is no more than $\ell + \mathsf{cc}_{\mathsf{PSM}}(N, \ell, k)$. □

4 PIR Encoding from Decomposable Matching Vectors

These two ingredients used to construct multi-party CDS, namely PIR encodings and the special-purpose PSM protocol are connected by the property of *decomposability*.

Definition 4.1 (k-decomposability). *Let $N' := \sqrt[k]{N}$. A family of vectors $(\mathbf{u}_i)_{i=1}^N$ is* k-decomposable *if there exist vector families $(\mathbf{u}_{1,i})_{i=1}^{N'}, \ldots, (\mathbf{u}_{k,i})_{i=1}^{N'}$ such that under the natural mapping $i \mapsto (i_1, \ldots, i_k) \in [N']^k$*

$$\mathbf{u}_i = \mathbf{u}_{1,i_1} \circ \ldots \circ \mathbf{u}_{k,i_k}$$

for all $i \in [N]$. Here, $\circ$ denotes the component-wise multiplication operation. A family of vectors $(\mathbf{u}_i)_{i\in[N]}$ is called a k-decomposable PIR encoding if it is a PIR encoding and it is k-decomposable.

In this section, we construct a k-decomposable (N, ℓ)-PIR encoding with $\ell = 2^{O(\sqrt{\log N} \log\log N)}$. In section 5, we show an efficient PSM for functionality (4) as long as $(\mathbf{u}_i)_{i\in[N]}$ is k-decomposable. Put together, they fulfill the assumptions in Theorem 3.1 and give us a communication-efficient multiparty CDS protocol.

4.1 PIR Encodings from Matching Vector Families

First, we define matching vector families and show that they give rise to PIR encodings. Our exposition here follows [LVW17] and uses techniques from [DG15].

Definition 4.2 (Matching vector family). *For integers N, ℓ, a collection of N pairs of vectors $\{(\mathbf{u}_i, \mathbf{v}_i)\}_{i=1}^N$ where all vectors are in $\mathbb{Z}_6^\ell$, is an* (N, ℓ)-matching vector family *if*

- *for any $i \in [N]$, $\langle \mathbf{u}_i, \mathbf{v}_i \rangle = 0$,*
- *for any $i \neq j \in [N]$, $\langle \mathbf{u}_i, \mathbf{v}_j \rangle \in \{1, 3, 4\}$.*

where all operations are done over $\mathbb{Z}_6$.

This definition is a specialization of the one from [Yek08, Efr12], and is sufficient for our purposes. The magical fact about matching vector families is that they exist for values of ℓ that are significantly less than N. (In contrast, if one replaces $\mathbb{Z}_6$ with $\mathbb{Z}_p$ for a prime p, we know that $\ell \geq N^{1/(p-1)}$ [BF98, BDL12]. It is thus a surprise that one can do much better when the modulus is a (small) composite number.

Lemma 4.3 ([Gro00]). *For every integer N, there exists an (N, ℓ)-matching vector family $\{(\mathbf{u}_i, \mathbf{v}_i)\}_{i \in [N]}$ of length $\ell = 2^{O(\sqrt{\log N \log \log N})} = N^{o(1)}$.*

We now show that any MV family gives rise to a PIR encoding scheme. This lemma was observed in the current form in [LVW17] and is implicit in the 2-server PIR protocol of [DG15].

Lemma 4.4 ([LVW17]). *If $\{(\mathbf{u}_i, \mathbf{v}_i)\}_{i=1}^N$ is an (N, ℓ)-matching vector family, then the family of vectors $\{1 \| \mathbf{u}_i\}_{i=1}^N$ is an $(N, \ell+1)$-PIR encoding.*

Proof. Define $\hat{\mathbf{u}}_i = 1 \| \mathbf{u}_i \in \mathbb{Z}_6^{\ell+1}$ and $\hat{\mathbf{v}}_i = 0 \| \mathbf{v}_i \in \mathbb{Z}_6^{\ell+1}$ for all $i \in [N]$. Then $\{(\hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i)\}_{i=1}^N$ remains an $(N, \ell+1)$-matching vector family since

$$\langle \hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i \rangle = \langle \mathbf{u}_i, \mathbf{v}_i \rangle.$$

Given a database $\mathbf{D} \in \{0,1\}^N$, an index $i \in [N]$, and randomness $\mathbf{w} \in \mathbb{Z}_6^\ell$, define auxiliary functions $G, G' : \{0,1\} \to \mathbb{Z}_6$, Following [DG15]. ($G$ and G' implicitly depend on i, $\mathbf{D}$ and $\mathbf{w}$). G and G' are defined as follows.

$$\begin{aligned} G(\mu) &:= \sum_{j \in [N]} \mathbf{D}[j] \cdot (-1)^{\langle \mu \hat{\mathbf{u}}_i + \mathbf{w}, \hat{\mathbf{v}}_j \rangle}, \text{ and} \\ G'(\mu) &:= \sum_{j \in [N]} \langle \hat{\mathbf{u}}_i, \hat{\mathbf{v}}_j \rangle \cdot \mathbf{D}[j] \cdot (-1)^{\langle \mu \hat{\mathbf{u}}_i + \mathbf{w}, \hat{\mathbf{v}}_j \rangle}. \end{aligned} \tag{5}$$

A straightforward computation shows the following (see [LVW17, Theorem 4.2]):

$$G'(1) - G(1) + G'(0) - G(0) = \mathbf{D}[i] \cdot (-1)^{\langle \mathbf{w}, \hat{\mathbf{v}}_i \rangle}. \tag{6}$$

For each $\mathbf{D} \in \{0,1\}^N$, define $H_{\mathbf{D}} : \mathbb{Z}_6^\ell \to \mathbb{Z}_6^\ell$ by

$$H_{\mathbf{D}}(\mathbf{z}) := (-1)^{\mathbf{z}[1]} \cdot \sum_{j \in [N]} (\hat{\mathbf{v}}_j - \mathbf{e}_1) \cdot \mathbf{D}[j] \cdot (-1)^{\langle \mathbf{z}, \hat{\mathbf{v}}_j \rangle}$$

where $\mathbf{e}_1 = (1, 0, \ldots, 0)$ is the first vector in the standard basis. Recalling that $\hat{\mathbf{u}}_i[1]$, the first bit of $\hat{\mathbf{u}}_i$, equals 1, we have

$$\begin{aligned}
\langle \hat{\mathbf{u}}_i, H_{\mathbf{D}}(\mu \hat{\mathbf{u}}_i + \mathbf{w}) \rangle &= (-1)^{\mu \hat{\mathbf{u}}_i[1] + \mathbf{w}[1]} \cdot \sum_{j \in [N]} \langle \hat{\mathbf{u}}_i, \hat{\mathbf{v}}_j - \mathbf{e}_1 \rangle \cdot \mathbf{D}[j] \cdot (-1)^{\langle \mu \hat{\mathbf{u}}_i + \mathbf{w}, \hat{\mathbf{v}}_j \rangle} \\
&= (-1)^{\mu + \mathbf{w}[1]} \cdot \sum_{j \in [N]} (\langle \hat{\mathbf{u}}_i, \hat{\mathbf{v}}_j \rangle - 1) \cdot \mathbf{D}[j] \cdot (-1)^{\langle \mu \hat{\mathbf{u}}_i + \mathbf{w}, \hat{\mathbf{v}}_j \rangle} \\
&= (-1)^{\mu + \mathbf{w}[1]} \cdot \Big(G'(\mu) - G(\mu) \Big).
\end{aligned}$$

Combined with equation (6), we see that:

$$\begin{aligned}
\langle \hat{\mathbf{u}}_i, H_{\mathbf{D}}(\hat{\mathbf{u}}_i + \mathbf{w}) \rangle - \langle \hat{\mathbf{u}}_i, H_{\mathbf{D}}(\mathbf{w}) \rangle &= (-1)^{1 + \mathbf{w}[1]} \cdot (G'(1) - G(1) + G'(0) - G(0)) \\
&= (-1)^{1 + \mathbf{w}[1]} \cdot \mathbf{D}[i] \cdot (-1)^{\langle \mathbf{w}, \hat{\mathbf{v}}_i \rangle} \\
&= \mathbf{D}[i] \cdot \phi_{\mathbf{w},i}.
\end{aligned}$$

where $\phi_{\mathbf{w},i} := (-1)^{1 + \mathbf{w}[1] + \langle \mathbf{w}, \hat{\mathbf{v}}_i \rangle}$. This completes the proof. □

4.2 Decomposable Matching Vector (DMV) Families

The main contribution of this section is the definition and construction of a decomposable matching vector family and thus, decomposable PIR encoding schemes.

Definition 4.5 (Decomposable Matching Vector Family). *For integers N, ℓ and $k \leq \log N$, a collection of vectors $\mathbf{u}_1, \ldots, \mathbf{u}_N, \mathbf{v}_1, \ldots, \mathbf{v}_N \in \mathbb{Z}_6^\ell$ is a* k-decomposable (N, ℓ)-matching vector family *if it is an (N, ℓ)-matching vector family and $(\mathbf{u}_i)_{i=1}^N$ is k-decomposable (as in definition 4.1).*

First, we show that decomposable matching vector families imply decomposable PIR encodings, extending Lemma 4.4.

Lemma 4.6. *For integers N, ℓ and $k \leq \log N$, if $\{(\mathbf{u}_i, \mathbf{v}_i)\}_{i=1}^N$ is a k-decomposable (N, ℓ)-matching vector family, then the family $\{1 \| \mathbf{u}_i\}_{i=1}^N$ is a k-decomposable $(N, \ell + 1)$-PIR encoding.*

Proof. By Lemma 4.4, $\{1 \| \mathbf{u}_i\}_{i=1}^N$ is a $(N, \ell + 1)$-PIR encoding.

Let $N' = \sqrt[k]{N}$ and let $\{\mathbf{u}_{1,i}\}_{i=1}^{N'}, \ldots, \{\mathbf{u}_{k,i}\}_{i=1}^{N'}$ be the k-decomposition of $\{\mathbf{u}_i\}_{i=1}^N$, then $\{1 \| \mathbf{u}_i\}_{i=1}^N$ is a k-decomposable and $\{1 \| \mathbf{u}_{1,i}\}_{i=1}^{N'}, \ldots, \{1 \| \mathbf{u}_{k,i}\}_{i=1}^{N'}$ is a k-decomposition of it. □

Thus, our main goal is to construct a decomposable matching vector family (whose parameters are slightly worse than that of the [Gro00] matching vector family). To this end, we build on the construction of MV families from the work of Grolmusz [Gro00].

Lemma 4.7 (implicit in [Gro00]). *For integers h, w, N and* any *distinct vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^h$ of Hamming weight exactly w. Then, there exists a matching vector family of N vectors where the vectors have length $\ell = h^{O(\sqrt{w})}$. In particular, the vectors $\mathbf{u}_i$ are indexed by subsets S of $[h]$ of size at most $O(\sqrt{w})$ and defined as follows:*

$$\mathbf{u}_i[S] = \prod_{j \in S} \mathbf{x}_i[j]$$

Proof. (Sketch.) Observe that for all $i, j \in [N]$:

$$\|\mathbf{x}_i \circ \mathbf{x}_j\|_1 \begin{cases} = w & \text{if } i = j \\ < w & \text{if } i \neq j \end{cases}$$

Let $\mathsf{thres}_w : \{0,1\}^h \to \mathbb{Z}_6$ denote the function which maps 0-1 vectors of Hamming weight exactly w to 0, and those of weight less than w to $\{1, 3, 4\}$. This means that

$$\begin{aligned} &\mathsf{thres}_w(\mathbf{x}_i \circ \mathbf{x}_i) = 0 \\ &\mathsf{thres}_w(\mathbf{x}_i \circ \mathbf{x}_j) \in \{1, 3, 4\} \quad \text{if } i \neq j \end{aligned}$$

The choices of $\mathbb{Z}_6$ and $\{1, 3, 4\}$ come from the work of Barrington, Beigel and Rudich [BBR94] which tells us that thres_w can be computed by a multilinear polynomial over $\mathbb{Z}_6$ of total degree $O(\sqrt{w})$.

Next, we will construct the vectors $\mathbf{u}_i$ and $\mathbf{v}_j$ of length $\ell = h^{O(\sqrt{w})}$ from $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively so that $\langle \mathbf{u}_i, \mathbf{v}_j \rangle = \mathsf{thres}_w(\mathbf{x}_i \circ \mathbf{x}_j)$ for all i, j. The bound on ℓ comes from the fact that we can write the evaluation of a multilinear polynomial of total degree $O(\sqrt{w})$ in h variables as the inner product of two vectors of length $\ell = h^{O(\sqrt{w})}$. In particular, $\mathbf{u}_i$ will be defined as above and $\mathbf{v}_j$ will be the coefficient vector of the degree $O(\sqrt{w})$ multilinear polynomial f_j which maps $\mathbf{x} \mapsto \mathsf{thres}_w(\mathbf{x} \circ \mathbf{x}_j)$. □

In our setting, the index $i = (i_1, \ldots, i_k) \in [\sqrt[k]{N}] \times \cdots \times [\sqrt[k]{N}]$ is divided amongst k players, as described in Section 2.1.

Lemma 4.8. *For integers N and $k \leq \log N$, there exists a k-decomposable (N, ℓ)-matching vectors family where $\ell = 2^{O(\sqrt{\log N} \cdot \log\log N)}$.*

Proof. Let h, w be multiples of k such that $\binom{h/k}{w/k}^k \geq N$. Let $\mathbf{y}_1, \ldots, \mathbf{y}_{\sqrt[k]{N}}$ be distinct vectors in $\{0,1\}^{h/k}$, each of Hamming weight w/k. For each $i = (i_1, \ldots, i_k) \in [N]$, we define

$$\mathbf{x}_i := \mathbf{y}_{i_1} \| \cdots \| \mathbf{y}_{i_k} \in \{0,1\}^h$$

Clearly, $\mathbf{x}_1, \ldots, \mathbf{x}_N$ is a collection of distinct vectors with Hamming weight w, and by Lemma 4.7, there exists a matching vector family $\{(\mathbf{u}_i, \mathbf{v}_i)\}_{i\in[N]}$, where the $\mathbf{u}_i$'s are indexed by subsets $S \subseteq [h]$ of size at most $O(\sqrt{w})$ and satisfies

$$\mathbf{u}_i[S] = \prod_{j\in S} \mathbf{x}_i[j] = \prod_{t=1}^{k} \prod_{j\in S_t} \mathbf{x}_i[j].$$

where $S_t := S \cap \{(t-1)\cdot \frac{h}{k} + 1, \ldots, t \cdot \frac{h}{k}\}$. Now, if we define

$$\mathbf{u}_{t,i_t}[S] := \prod_{j\in S_t} \mathbf{x}_i[j] = \prod_{j\in S_t} \mathbf{y}_{i_t}[j - (t-1)\cdot \tfrac{h}{k}],$$

we have $\mathbf{u}_i = \mathbf{u}_{1,i_1} \circ \ldots \circ \mathbf{u}_{k,i_k}$, giving us a k-decomposition.

Set $w = \lceil \log N/k \rceil \cdot k$ and $h = 2w$ (so that h, w are multiples of k). Then,

$$\binom{h/k}{w/k}^k \geq ((h/w)^{w/k})^k = 2^w \geq N$$

Also, we have $\ell = h^{O(\sqrt{w})} = 2^{O(\sqrt{\log N}\cdot \log\log N)}$. □

As a result, we have the main theorem of this section:

Theorem 4.9. *For integers N and $k \leq \log N$, there exists a k-decomposable (N, ℓ)-PIR encoding where $\ell = 2^{O(\sqrt{\log N}\cdot \log\log N)}$.*

5 A Special-Purpose PSM Protocol

Given the decomposable PIR encoding from Section 4.2, the final piece required to instantiate the framework in Section 3 is a special purpose PSM protocol for the functionality described in Equation 4.

Theorem 5.1. *For integers N, ℓ and $k \leq \log N$, if $(\mathbf{u}_i)_{i=1}^N$ is k-decomposable, then there is a PSM for the functionality*

$$\begin{aligned} F_{\mathsf{aux}} &: [\sqrt[k]{N}] \times \ldots \times [\sqrt[k]{N}] \times (\{0,1\} \times \mathbb{Z}_6^\ell \times \mathbb{Z}_6^\ell) \to \mathbb{Z}_6^\ell \times \mathbb{Z}_6 \\ \text{where } F_{\mathsf{aux}}&(i_1, \ldots, i_k; (\mu, \mathbf{w}, \mathbf{r})) \mapsto (\mathbf{w} + \mu\mathbf{u}_i, \langle \mathbf{u}_i, \mathbf{r}\rangle) \end{aligned} \tag{4}$$

with communication complexity $O(\ell k^2)$ per party.

In order to construct a efficient PSM protocol for this specialized functionality, we show that (a) this functionality can be written as an affine mod-6 branching program of size $O(k\cdot\ell)$ and (b) use the fact that there are efficient PSM protocols that compute affine branching programs over rings [IK00, IK02, CFIK03] where the total communication is polynomial in the size of the branching program.

Lemma 5.2. *There is a PSM protocol for the k-party affine branching program functionality $\mathbf{BP}^k_m$ over the ring $\mathbb{Z}_6$ with communication complexity $O(m^2)$ per party.*

Proof. A mod-$\mathbb{Z}_6$ branching program can be locally decomposed into a mod-$\mathbb{F}_2$ branching program and a mod-$\mathbb{F}_3$ branching program using Chinese remainder theorem. [IK02] show a $\mathsf{cc} = O(m^2 \log p)$ PSM protocol for mod-$\mathbb{F}_p$ branching program when p is a prime.

Moreover, [IK02] can be immediately extended to branching programs over any commutative rings; see Appendix A for more details. It is further extended by [CFIK03] to branching programs over any rings. □

Proof (Proof of Theorem 5.1). Let $\{\mathbf{u}_{1,i}\}_{i=1}^{\sqrt[k]{N}}, \ldots, \{\mathbf{u}_{k,i}\}_{i=1}^{\sqrt[k]{N}} \subseteq \mathbb{Z}_6^\ell$ be the decomposition, then under the natural mapping $i \mapsto (i_1, \ldots, i_N) \in [\sqrt[k]{N}]^k$, we have

$$\mathbf{u}_i = \mathbf{u}_{1,i_1} \circ \ldots \circ \mathbf{u}_{k,i_k}$$

for all $i \in [N]$.

For the functionality

$$F^{(1)}_{\mathsf{aux}}(i_1, \ldots, i_k; \mu, \mathbf{w}) = \mu \mathbf{u}_i + \mathbf{w} \in \mathbb{Z}_6^\ell,$$

its j-th output bit can be written as

$$(\mu \mathbf{u}_i + \mathbf{w})[j] = \mu \mathbf{u}_{1,i_1}[j] \mathbf{u}_{2,i_2}[j] \ldots \mathbf{u}_{k,i_k}[j] + \mathbf{w}[j].$$

Since the t-th party ($1 \le t \le k$) can compute $\mathbf{u}_{t,i_t}$ locally, the j-th bit of $\mu \mathbf{u}_i + \mathbf{w}$ can be computed by a mod-$\mathbb{Z}_6$ branching program with $O(k)$ nodes. By Lemma 5.2, there exists a PSM scheme evaluating the j-th output bit using $O(k^2)$ communication per party. Thus, all ℓ outputs can be computed with communication complexity $O(\ell k^2)$ per party.

For the functionality

$$F^{(2)}_{\mathsf{aux}}(i_1, \ldots, i_k; \mathbf{r}) = \langle \mathbf{u}_i, \mathbf{r} \rangle \in \mathbb{Z}_6,$$

let $\mathbf{s} \in \mathbb{Z}_6^\ell$ be a random vector sampled from CRS such that $\sum_{j=1}^{\ell} \mathbf{s}[j] = 0$. Then, instead of computing the functionality, the parties compute together the functionality

$$F^{(2)'}_{\mathsf{aux}}(i_1, \ldots, i_k; \mathbf{r}, \mathbf{s}) = \mathbf{u}_i \circ \mathbf{r} + \mathbf{s} \in \mathbb{Z}_6^\ell,$$

It is easy to see that $\mathbf{u}_i \circ \mathbf{r} + \mathbf{s}$ reveals $\langle \mathbf{u}_i, \mathbf{r} \rangle$ and nothing more. The j-th bit of $\mathbf{u}_i \circ \mathbf{r} + \mathbf{s}$ can be written as

$$(\mathbf{u}_i \circ \mathbf{r} + \mathbf{s})[j] = \mathbf{u}_{1,i_1}[j] \mathbf{u}_{2,i_2}[j] \ldots \mathbf{u}_{k,i_k}[j] \mathbf{r}[j] + \mathbf{s}[j],$$

which can again be computed by a simple affine branching program with $O(k)$ nodes. By Lemma 5.2, there exists a PSM scheme evaluating this with $O(k^2)$ bits of communication per party. Thus, to compute all the bits, we need $O(\ell k^2)$ bits of communication.

Clearly, once Charlie learns $\mathbf{u}_i \circ \mathbf{r} + \mathbf{s}$, he can add up all the bits to get $\langle \mathbf{u}_i, \mathbf{r} \rangle$. □

6 Putting Together

We finally put together all the pieces to construct a multiparty CDS scheme, and various types of secret sharing schemes.

6.1 Multi-party CDS for $\mathbf{INDEX}_N^{k+1}$

We obtain the multiparty CDS protocol by instantiating our general framework in Section 3 with the decomposable PIR encodings in Section 4.2 and the PSM protocol in Section 5.

Theorem 6.1. *For $1 \leq k \leq \log N$, there is a $(k+1)$-party CDS protocol for $\mathbf{INDEX}_N^{k+1}$ whose communication complexity is $2^{O(\sqrt{\log N} \log\log N)}$.*

Proof. Theorem 3.1 gives us a $k+1$-party CDS protocol for $\mathbf{INDEX}_N^{k+1}$ assuming a k-decomposable PIR encoding scheme and a PSM protocol for the associated functionality F_{aux}. Theorem 4.9 constructs such a k-decomposable (N,ℓ)-PIR encoding scheme with $\ell = 2^{O(\sqrt{\log N}\log\log N)}$ and Theorem 5.1 constructs a PSM protocol for the associated functionality F_{aux} with communication complexity $O(k^2 \cdot \ell)$. Put together, applying Theorem 3.1, we get a $(k+1)$-party CDS protocol with communication complexity $O(\ell + k^2\ell) = 2^{O(\sqrt{\log N}\log\log N)}$. □

Because CDS for $\mathbf{ALL}_N^k$ is easier than CDS for $\mathbf{INDEX}_N^{k+1}$, we also get:

Corollary 6.2 *For $1 \leq k \leq \log N$, there is a k-party CDS protocol for $\mathbf{ALL}_N^k$ whose communication complexity is $2^{O(\sqrt{\log N}\log\log N)}$.*

6.2 From CDS to Non-Monotone Secret Sharing

A non-monotone secret-sharing scheme for access function F is a randomized algorithm that on input a secret bit, outputs n pairs of shares $(s_{i,0}, s_{i,1})_{i\in[n]}$ such that for any $(x_1,\ldots,x_n) \in \{0,1\}^n$, the n shares $(s_{1,x_1},\ldots,s_{n,x_n})$ determine the secret if $F(x_1,\ldots,x_n) = 1$ and reveal nothing about the secret otherwise.

Definition 6.3 (Non-monotone Secret Sharing). *Given a function $F : \{0,1\}^n \to \{0,1\}$, a non-monotone secret-sharing scheme for access function F is a randomized algorithm*

$$\mathsf{nmSS} : \mathcal{M} \times \mathcal{W} \to (\{0,1\}^{\mathsf{cc}})^{2n}$$

that on input a secret bit, outputs n pairs of shares $s_{1,0}, s_{1,1}, \ldots, s_{n,0}, s_{n,1} \in \{0,1\}^{\mathsf{cc}}$ satisfying the following properties:

(correctness.) *There exists a reconstruction algorithm $\mathsf{C} : \{0,1\}^n \times (\{0,1\}^{\mathsf{cc}})^n \to \mathcal{M}$ such that for all $(x_1,\ldots,x_n) \in \{0,1\}^n$ that $F(x_1,\ldots,x_n) = 1$ and for all $\mu \in \mathcal{M}, w \in \mathcal{W}$,*

$$\mathsf{nmSS}(\mu; w) = (s_{1,0}, s_{1,1}, \ldots, s_{n,0}, s_{n,1}) \implies \mathsf{C}(x_1,\ldots,x_n, s_{1,x_1},\ldots,s_{n,x_n}) = \mu.$$

(privacy.) *There exists a simulator* S *such that for all* $(x_1, \ldots, x_n) \in \{0,1\}^n$ *satisfying* $F(x_1, \ldots, x_n) = 0$*, the joint distribution of* $(s_{1,x_1}, \ldots, s_{n,x_n})$ *is perfectly indistinguishable from* $\mathsf{S}(x_1, \ldots, x_k)$*, where* $(s_{1,0}, s_{1,1}, \ldots, s_{n,0}, s_{n,1}) := \mathsf{nmSS}(\mu; w)$ *and the randomness are taken over* $w \xleftarrow{R} \mathcal{W}$ *and the coin tosses of* S*.*

Standard monotone secret-sharing correspond to the special case where F is monotone and $s_{1,0} = \cdots = s_{n,0} = \perp$. In such case, let $s_1, \ldots, s_n$ denotes $s_{1,1}, \ldots, s_{n,1}$ respectively.

Let $N := 2^n$. It is not hard to see that non-monotone secret sharing for all n-party access functions F is the same as n-party CDS for $\mathbf{ALL}_N^n$. This connection is almost syntactic, and can be formalized as follows.

Lemma 6.4. *For any access function* $F : \{0,1\}^n \to \{0,1\}$*, there is a non-monotone secret-sharing scheme for* F *with share size* cc *if and only if there is a CDS scheme for* $\mathbf{ALL}_{2^n}^n$ *with communication complexity* cc *when the predicate is* F*.*

Proof. Assume $(\mathsf{B}_1, \ldots, \mathsf{B}_n)$ is a CDS for $\mathbf{ALL}_{2^n}^n$ with predicate function F, a non-monotone secret sharing scheme for F can be defined as

$$\mathsf{nmSS}(\mu; w) = (\mathsf{B}_1(\mu, 0; w), \mathsf{B}_1(\mu, 1; w), \ldots, \mathsf{B}_n(\mu, 0; w), \mathsf{B}_n(\mu, 1; w)).$$

Assume nmSS is a non-monotone secret sharing scheme for access function F. Then $(\mathsf{B}_1, \ldots, \mathsf{B}_n)$ is a CDS for $\mathbf{ALL}_{2^n}^n$ with predicate function F if $\mathsf{B}_t(x_t; w)$ outputs s_{t,x_t} and s_{t,x_t} is defined as $(s_{1,0}, s_{1,1}, \ldots, s_{n,0}, s_{n,1}) := \mathsf{nmSS}(\mu; w)$. □

Thus, we obtain the following theorem, disproving Conjecture 2.

Theorem 6.5. *There is a non-monotone secret-sharing for the family of all access functions* $F : \{0,1\}^n \to \{0,1\}$ *with total share size* $2^{O(\sqrt{n} \log n)}$ *bits.*

6.3 From Non-Monotone to Monotone Secret Sharing

We describe the following folklore transformation which transforms a *non-monotone* secret-sharing scheme for any $F : \{0,1\}^n \to \{0,1\}$ into a *monotone* secret-sharing scheme with roughly the same share size for a related function F'.

Lemma 6.6. *For any function* $F : \{0,1\}^n \to \{0,1\}$*, a non-monotone secret sharing scheme for* F *with share size* cc *implies a monotone secret sharing scheme for a monotone access function* $F' : \{0,1\}^{2n} \to \{0,1\}$ *with share size* $\mathsf{cc} + 2$*, where* $F' : \{0,1\}^{2n} \to \{0,1\}$ *is defined as*

$$F'(x'_1, \ldots, x'_{2n}) = \begin{cases} 1 & \text{if } \exists i \text{ such that } x'_{2i-1} = x'_{2i} = 1 \\ 0 & \text{else if } \exists i \text{ such that } x'_{2i-1} = x'_{2i} = 0 \\ F(x'_2, x'_4, \ldots, x'_{2n}) & \text{otherwise.} \end{cases}$$

Proof. It is easy that F' is monotone. Let nmSS be the non-monotone secret-sharing scheme for F, we construct a monotone secret sharing scheme mSS for F' as follows:

On input $\mu \in \{0,1\}$
1. Sample bits $b_1, \ldots, b_n, r_1, \ldots, r_n$ and $w \in \mathcal{W}$ uniformly at random.
2. Let $(s_{1,0}, s_{1,1}, \ldots, s_{n,0}, s_{n,1}) := \mathsf{nmSS}(\mu \oplus r_1 \oplus \ldots \oplus r_n; w)$.
3. Output $(s'_1, \ldots, s'_{2n})$ where

$$s'_{2i-1} := (s_{i,0}, r_i, b_i) \qquad s'_{2i} := (s_{i,1}, r_i, b_i \oplus \mu).$$

The reconstruction algorithm for mSS either computes μ from $(b_i, b_i \oplus \mu)$, or runs the one for nmSS to recover $\mu \oplus r_1 \oplus \cdots \oplus r_n$ and thus μ. More precisely, we argue correctness and privacy via a case analysis. For any $(x'_1, \ldots, x'_{2n}) \in \{0,1\}^{2n}$, given the shares $(s_j)_{x'_j=1}$:

Case 1: $\exists i$ *s.t.* $(x'_{2i-1}, x'_{2i}) = (1,1)$. Here, $F'(x'_1, \ldots, x'_{2n}) = 1$, and the reconstruction algorithm can recover μ from $b_i, b_i \oplus \mu$ given in s'_{2i-1}, s'_{2i}.

Case 2: $\exists i$ *s.t.* $(x'_{2i-1}, x'_{2i}) = (0,0)$. Here, $F'(x'_1, \ldots, x'_{2n}) = 0$, and privacy follows from the fact that r_i is perfectly hidden, and therefore μ is also perfectly hidden.

Case 3: $\forall i, (x'_{2i-1}, x'_{2i}) \in \{(0,1),(1,0)\}$. So $F'(x'_1, \ldots, x'_{2n}) = F(x'_2, x'_4, \ldots, x'_{2n})$. First, observe that the shares $(s_j)_{x'_j=1} = (s'_{2i-1+x'_{2i}})_{i \in [n]}$ and reveal exactly

$$s_{1,x'_2}, s_{2,x'_4}, \ldots, s_{n,x'_{2n}}, r_1, \ldots, r_n.$$

Now, if $F(x'_2, x'_4, \ldots, x'_{2n}) = 1$, correctness of nmSS enables Charlie to recover $\mu \oplus r_1 \oplus \ldots \oplus r_n$ and thus μ. Otherwise, privacy of nmSS hides $\mu \oplus r_1 \oplus \ldots \oplus r_n$, thus μ is perfectly hidden.

This completes the proof. □

As for non-monotone secret sharing, there are double exponential different access functions. Under the mapping specified in Lemma 6.6, they are mapped to different monotone functions. Thus, we obtain the following theorem, disproving Conjecture 1.

Theorem 6.7. *There is a collection $\hat{F}_n$ of $2^{2^{n/2}}$ monotone functions over $\{0,1\}^n$, such that there is monotone secret sharing scheme for $\hat{F}_n$ with share size $2^{O(\sqrt{n}\log n)}$ bits.*

To obtain the more general statement in Theorem 1 (informal), we just apply the above construction to the first $\log s$ bits of the input.

7 Linear CDS and Secret Sharing

A CDS protocol is linear if the transmitting functions are linear on the secret and randomness (not necessarily linear on the inputs). A secret sharing scheme is linear if the share generation algorithm is linear on the secret and randomness.

We present a linear non-monotone secret sharing scheme (equivalently, a multiparty CDS protocol) for every access function over $\{0,1\}^n$ with shares of size $O(2^{n/2})$. It is sufficient to construct a linear $(n+1)$-party CDS for $\mathbf{INDEX}_{2^n}^{n+1}$ where each party sends $O(2^{n/2})$ bits. The construction will follow our general framework for building multi-party CDS from 2-party CDS.

In CDS for $\mathbf{INDEX}_{2^n}^{n+1}$, each of the first n parties holds a bit of $i \in \{0,1\}^n$, the last party holds a truth-table $\mathbf{D} \in \{0,1\}^{2^n}$, the secret is disclosed if and only if $\mathbf{D}[i] = 1$. As a warm-up, we recap the 2-party linear CDS for $\mathbf{INDEX}_{2^n}^2$ with total communication $O(2^{n/2})$.

2-party CDS [GKW15]. Bob holds $i \in \{0,1\}^n$ and split it into higher half $j^H \in \{0,1\}^{n/2}$ and lower half $j^L \in \{0,1\}^{n/2}$. The shared randomness is $\mathbf{w}, \mathbf{r} \in \{0,1\}^{2^{n/2}}$. Bob sends

$$\mathbf{m}_B^1 := \mu \mathbf{e}_{j^L} + \mathbf{w} \in \{0,1\}^{2^{n/2}}, \qquad m_B^2 := \mathbf{r}[j^H]$$

to Charlie. Alice holds the truthtable $\mathbf{D} \in \{0,1\}^{2^n}$ which can be viewed as a $2^{n/2} \times 2^{n/2}$ matrix, sends

$$\mathbf{m}_A := \mathbf{D}\mathbf{w} + \mathbf{r} \in \{0,1\}^{2^{n/2}}.$$

Charlie computes

$$\begin{aligned} & \mathbf{e}_{j^H} \cdot \mathbf{D} \cdot \mathbf{m}_B^1 - \mathbf{e}_{j^H} \cdot \mathbf{m}_A + m_B^2 \\ = & \mathbf{e}_{j^H} \cdot \mathbf{D} \cdot (\mu \mathbf{e}_{j^L} + \mathbf{w}) - \mathbf{e}_{j^H} \cdot (\mathbf{D}\mathbf{w} + \mathbf{r}) + \mathbf{r}[j^H] \\ = & \mu \mathbf{e}_{j^H} \cdot \mathbf{D} \cdot \mathbf{e}_{j^L} \\ = & \mu \mathbf{D}[i]. \end{aligned} \tag{7}$$

PSM building block. Following our general framework for building multi-party CDS from 2-party CDS, the crux is to construct a "linear" n-party PSM for the following functionality where each party sends $O(2^{n/2})$ bits.

$$(j^H; \mathbf{r}) \mapsto \mathbf{r}[j^H] \in \{0,1\} \tag{8}$$

$$(j^L; \mathbf{w}, \mu) \mapsto \mu \mathbf{e}_{j^L} + \mathbf{w} \in \{0,1\}^{2^{n/2}} \tag{9}$$

where $\mathbf{w} \in \{0,1\}^{2^{n/2}}$. For this, we should use "partial garbling" [IW14] and exploit the fact that the PSM does not need to protect the privacy of j^H, j^L, which we may treat as "public" inputs. In addition, observe that the computation in (8) and (9) are linear on the "private" inputs $\mathbf{r}, \mathbf{w}, \mu$. Our goal is a protocol where

– the communication of the "partial garbling PSM" is $O(2^{n/2})$.
– each party's PSM message is linear in $\mathbf{r}, \mathbf{w}, \mu$ and the PSM randomness.

As a partial PSM protocol for functionality (8): Sample random vectors $\mathbf{r}_1, \ldots, \mathbf{r}_{n/2} \in \{0,1\}^{2^{n/2}}$ such that $\mathbf{r} = \sum_{t=1}^{n/2} \mathbf{r}_t$. For every $1 \le t \le n/2$, the party who holds j_t^H sends $\mathbf{r}_t[j]$ for all $j \in \{0,1\}^{n/2}$ such that $j_t = j_t^H$. Charlie can reconstruct $\mathbf{r}[j^H] = \sum_t \mathbf{r}_t[j^H]$.

As a partial PSM protocol for functionality (9): For every $1 \le t \le n/2$, the party who holds j_t^L sends $\mathbf{w}[j]$ for all $j \in \{0,1\}^{n/2}$ such that $j_t \neq j_t^L$. One party sends $\mu + \sum_j \mathbf{w}[j]$ in addition. Charlie can reconstruct vector $\mu \mathbf{e}_{j^L} + \mathbf{w}$ as he receives all the bits of it except the j^L-th bit and the j^L-th bit can be recovered from the sum the vector and the rest of the vector.

CDS for $\mathbf{INDEX}_N^{k+1}$

Input of P_t ($1 \le t \le k$): $i_t \in [\sqrt[k]{N}]$ and $\mu \in \{0,1\}$.
Input of P_{k+1} (Alice): $\mathbf{D} \in \{0,1\}^{\sqrt{N} \times \sqrt{N}}$.
Shared Randomness: $\mathbf{w}, \mathbf{r}_{k/2+1}, \ldots, \mathbf{r}_k \in \{0,1\}^{\sqrt{N}}$

The protocol proceeds as follows.

– For $1 \le t \le k/2$, the t-th party sends $\mathbf{w}[j]$ for all $j \in [\sqrt[k]{N}]^{k/2}$ such that $j_t \neq i_t$.
– For $k/2+1 \le t \le k$, the t-th party sends $\mathbf{r}_t[j]$ for all $j \in [\sqrt[k]{N}]^{k/2}$ such that $j_{t-k/2} = i_t$.
– P_{k+1} (Alice) sends $\mathbf{m}_A := \mathbf{D}\mathbf{w} + \sum_t \mathbf{r}_t$.
– In addition, one of parties sends $m_D := \mu + \sum_j \mathbf{w}[j]$.
– Charlie computes $\mathbf{w}' := \mathbf{w} + \mu \mathbf{e}_{i^L}$ by the following:
For every $j \neq i^L$, $\mathbf{w}'[j] = \mathbf{w}[j]$ and $\mathbf{w}[j]$ is sent by one of the first $k/2$ parties. And $\mathbf{w}'[i^L] := m_D - \sum_{j \neq i^L} \mathbf{w}'[j]$.
– Charlie outputs 1 if $\mathbf{e}_{i^H} \cdot D \cdot \mathbf{w}' - \mathbf{m}_A[i^H] + \sum_t \mathbf{r}_t[i^H] = 1$, and outputs 0 otherwise.

Fig. 2. $(k+1)$-party linear CDS for $\mathbf{INDEX}_N^{k+1}$, when k is even.

Theorem 7.1. *For even $k \le \log N$, there is a $(k+1)$-party linear CDS protocol for $\mathbf{INDEX}_N^{k+1}$ (Figure 2) whose communication complexity is $O(\sqrt{N})$ per party.*

Proof. The argument for correctness is the same as that for 2-party CDS: Let $\mathbf{r} := \sum_t \mathbf{r}_t$. Charlie learns $\mathbf{w}' = \mu \mathbf{e}_{j^L} + \mathbf{w}$ from the first $k/2$ parties, learns $\mathbf{r}[j^H]$ from the following $k/2$ parties, gets $\mathbf{D}\mathbf{w} + \mathbf{r}$ from the last party (Alice), then he can reconstruct $\mu \cdot \mathbf{D}[i]$ using equation (7).

Privacy follows from the following observations:

- When $\mathbf{D}[i] = 0$, the joint distribution of $\mathbf{w}', \mathbf{m}_A, \mathbf{r}[i^H]$ can be simulated from $\mathbf{D}, i$ without knowing μ. This is due the security of 2-party CDS for **INDEX** [GKW15].
- The messages from the first $k/2$ parties together with m_D can be determined by $(\mathbf{w}', j^L)$: the messages from the first $k/2$ parties all bits in $\mathbf{w}'$; and $m_D = \sum_j \mathbf{w}'[j]$.
- The messages from the following $k/2$ parties can be simulated from $(\mathbf{r}[j^H], j^H)$: sample $\mathbf{r}_{k/2+1}, \ldots, \mathbf{r}_k$ such that $\sum_t \mathbf{r}_t[j^H] = \mathbf{r}[j^H]$; the t-th party sends $\mathbf{r}_t[j]$ for all $j \in [\sqrt[k]{N}]^{k/2}$ such that $j_{t-k/2} = i_t$.

Theorem 7.2. *There is a linear non-monotone secret-sharing for the family of all access functions $F : \{0,1\}^n \to \{0,1\}$ with share size $O(2^{n/2})$ bits for each party.*

Proof. (Sketch) Use Lemma 6.4 to construct non-monotone secret sharing from multi-party CDS for **ALL**, with the observation that the transformation in Lemma 6.4 preserves linearity.

Acknowledgments. We thank Yuval Ishai for telling us about Conjecture 1. We thank the anonymous EUROCRYPT 2018 reviewers for their insightful comments.

References

BBR94. David A. Mix Barrington, Richard Beigel, and Steven Rudich. Representing boolean functions as polynomials modulo composite numbers. *Computational Complexity*, 4:367–382, 1994.

BDL12. Abhishek Bhowmick, Zeev Dvir, and Shachar Lovett. New lower bounds for matching vector codes. *CoRR*, abs/1204.1367, 2012.

Bei11. Amos Beimel. *Secret-Sharing Schemes: A Survey*, pages 11–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

BF98. Laszlo Babai and Peter Frankl. Linear algebra methods in combinatorics, 1998.

BGP95. Amos Beimel, Anna Gál, and Mike Paterson. Lower bounds for monotone span programs. In *FOCS*, pages 674–681, 1995.

BGW99. László Babai, Anna Gál, and Avi Wigderson. Superpolynomial lower bounds for monotone span programs. *Combinatorica*, 19(3):301–319, 1999.

BI01. Amos Beimel and Yuval Ishai. On the power of nonlinear secret-sharing. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 188–202. IEEE Computer Society, 2001.

BIKK14. Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In *TCC*, pages 317–342, 2014.

BL88. Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 27–35, 1988.

Bla79. George Robert Blakley. Safeguarding cryptographic keys. In *Proc. AFIPS 1979 National Computer Conference*, pages 313–317, 1979.

CFIK03. Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In *EUROCRYPT*, pages 596–613, 2003.

CKGS98. Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

Csi97. László Csirmaz. The size of a share must be large. *J. Cryptology*, 10(4):223–231, 1997.

DG15. Zeev Dvir and Sivakanth Gopi. 2-server PIR with sub-polynomial communication. In *STOC*, pages 577–584, 2015.

DGY11. Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching vector codes. *SIAM J. Comput.*, 40(4):1154–1178, 2011.

Efr12. Klim Efremenko. 3-query locally decodable codes of subexponential length. volume 41, pages 1694–1703, 2012.

FKN94. Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 554–563. ACM, 1994.

GIKM00. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *J. Comput. Syst. Sci.*, 60(3):592–629, 2000.

GKW15. Romain Gay, Iordanis Kerenidis, and Hoeteck Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In *CRYPTO (II)*, pages 485–502, 2015.

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

Gro00. Vince Grolmusz. Superpolynomial size set-systems with restricted intersections mod 6 and explicit ramsey graphs. *Combinatorica*, 20(1):71–86, 2000.

IK97. Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *ISTCS*, pages 174–184, 1997.

IK00. Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304. IEEE Computer Society, 2000.

IK02. Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, pages 244–256, 2002.

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

IW14. Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 650–662. Springer, 2014.

KW93. Mauricio Karchmer and Avi Wigderson. On span programs. In *Structure in Complexity Theory Conference*, pages 102–111. IEEE Computer Society, 1993.
LVW17. Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Conditional disclosure of secrets via non-linear reconstruction. In *CRYPTO Part I*, pages 758–790, 2017.
OSW07. Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203, 2007.
PR17. Toniann Pitassi and Robert Robere. Lifting nullstellensatz to monotone span programs over any field. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:165, 2017.
RPRC16. Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook. Exponential lower bounds for monotone span programs. In *FOCS*, pages 406–415, 2016.
Sha79. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
VV15. Vinod Vaikuntanathan and Prashant Nalini Vasudevan. Secret sharing and statistical zero knowledge. In *ASIACRYPT (I)*, pages 656–680, 2015.
WY05. David P. Woodruff and Sergey Yekhanin. A geometric approach to information-theoretic private information retrieval. In *CCC*, pages 275–284, 2005.
Yek08. Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 55(1):1:1–1:16, 2008.

A Private Simultaneous Message for Branching Program over Commutative Rings

We sketch the PSM schemes from [IK02, CFIK03] for branching programs over commutative rings.

PSM for $\mathbf{AFFINE}^k_m$

t-th Party's Input: $\mathbf{x}_t \in \mathcal{R}^n$.
Shared Input: Affine function $f : \mathcal{R}^{nk} \to \mathcal{R}$ such that

$$f_{\mathbf{y}_1,\ldots,\mathbf{y}_k,c}(\mathbf{z}_1,\ldots,\mathbf{z}_k) := \langle \mathbf{z}_1, \mathbf{y}_1 \rangle + \ldots + \langle \mathbf{z}_k, \mathbf{y}_k \rangle + c$$

Shared Randomness: $r_1, \ldots, r_{k-1} \in \mathcal{R}$.

- The t-th party (for $1 \le t < k$) sends $m_t := \langle \mathbf{x}_t, \mathbf{y}_t \rangle + r_t$
- The k-th party sends $m_k := \langle \mathbf{x}_k, \mathbf{y}_k \rangle + c - \sum_{t=1}^{k-1} r_t$,
- Charlie outputs $\sum_{t=1}^{k} m_t$.

Fig. 3. The $\log |\mathcal{R}|$-bit PSM protocol for $\mathbf{AFFINE}^k$.

Theorem A.1 (folklore). *There is a PSM scheme (Figure 3) for* $\mathbf{AFFINE}^k$ *over commutative ring* $\mathcal{R}$ *such that every party sends one ring element.*

Proof. The correctness is straight-forward,

$$\sum_{t=1}^{k} m_t = \sum_{t=1}^{k-1}(\langle \mathbf{x}_t, \mathbf{y}_t\rangle + r_t) + \langle \mathbf{x}_k, \mathbf{y}_k\rangle + c - \sum_{t=1}^{k-1} r_t = \sum_{t=1}^{k}\langle \mathbf{x}_t, \mathbf{y}_t\rangle + c = f(\mathbf{x}_1, \ldots, \mathbf{x}_k).$$

Privacy follows from the following observations:

- the joint distribution of $m_1, \ldots, m_{k-1}$ is uniformly random, since we are using $(r_1, \ldots, r_{k-1})$ is one-time pads;
- m_k is determined by $m_1, \ldots, m_{k-1}$ and $f(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ as $m_k = f(\mathbf{x}_1, \ldots, \mathbf{x}_k) - \sum_{t=1}^{k-1} m_t$.

Putting the two together, we can simulate Charlie's view given just $f(\mathbf{x}_1, \ldots, \mathbf{x}_k)$. □

Lemma A.2 ([IK02]). *For any matrix* $M \in \mathcal{R}^{m\times m}$ *such that* $M_{i,j} = -1$ *for* $i = j+1$ *and* $M_{i,j} = 0$ *for* $i > j+1$*, there exist matrix* $L, R \in \mathcal{R}^{m\times m}$ *satisfying* (10) *such that*

$$M = \underbrace{\begin{bmatrix} 1 & L_{1,2} & \cdots & L_{1,m} \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}}_{L} \cdot \begin{bmatrix} & & & \det M \\ -1 & & & \\ & \ddots & & \\ & & -1 & \end{bmatrix} \cdot \underbrace{\begin{bmatrix} 1 & R_{1,2} & \cdots & R_{1,m} \\ & 1 & \ddots & \vdots \\ & & \ddots & R_{m-1,m} \\ & & & 1 \end{bmatrix}}_{R}$$

Lemma A.3 ([IK02]). *For any matrix* $M \in \mathcal{R}^{m\times m}$ *such that* $M_{i,j} = -1$ *for* $i = j+1$ *and* $M_{i,j} = 0$ *for* $i > j+1$*, the distribution of* $L \cdot M \cdot R$ *is determined by* $\det M$*, where* $L, R \in \mathcal{R}^{m\times m}$ *are random matrices satisfying* (10).

Proof. Let L_M, R_M be the matrices implied by Lemma A.2 such that L_M, R_M are upper triangular matrices with 1's in their diagonal, $L^{(M)}_{i,j} = 0$ for $1 < i < j \le m$ and

$$M = L_M \cdot \begin{bmatrix} & & & \det M \\ -1 & & & \\ & \ddots & & \\ & & -1 & \end{bmatrix} \cdot R_M$$

$L \cdot M \cdot R$ can be written as

$$L \cdot M \cdot R = \underbrace{L \cdot L_M}_{\text{same distribution as } L} \cdot \begin{bmatrix} -1 & & & \det M \\ & \ddots & & \\ & & -1 & \end{bmatrix} \cdot \underbrace{R_M \cdot R}_{\text{same distribution as } R}$$

The joint distribution of $(L \cdot L_M, R_M \cdot R)$ is the same as (L, R), thus $\det M$ determines the distribution of $L \cdot M \cdot R$. □

PSM for $\mathbf{BP}^k$

t-th Party's Input: $\mathbf{x}_t \in \mathcal{R}^n$.
Shared Input: Affine functions $F_{i,j} : \mathcal{R}^{nk} \to \mathcal{R}$ for $1 \le i \le j \le m$.
Shared Randomness: $\{L_{1,j}\}_{j=2}^{m} \in \mathcal{R}^{m-1}$, $\{R_{i,j}\}_{1 \le i < j \le m} \in \mathcal{R}^{m(m-1)/2}$ and randomness for $\frac{1}{2}m(m+1)$ nested PSM runs for $\mathbf{AFFINE}^k$

- Define matrix L, R such that

$$L_{i,j} = \begin{cases} 1, & \text{if } i = j \\ L_{1,j}, & \text{if } 1 = i < j \\ 0, & \text{otherwise} \end{cases} \qquad R_{i,j} = \begin{cases} 1, & \text{if } i = j \\ R_{i,j}, & \text{if } i < j \\ 0, & \text{if } i > j \end{cases} \tag{10}$$

  Each of the k parties independently computes affine functions $F'_{i,j} : \mathcal{R}^{nk} \to \mathcal{R}$ for $1 \le i \le j \le m$ defined as

$$F'(\mathbf{z}_1, \ldots, \mathbf{z}_n) = L \cdot F(\mathbf{z}_1, \ldots, \mathbf{z}_n) \cdot R$$

  where $F' : \mathcal{R}^{nk} \to \mathcal{R}^{m \times m}$ is defined as

$$\big(F'(\mathbf{z}_1, \ldots, \mathbf{z}_k)\big)_{i,j} = \begin{cases} F'_{i,j}(\mathbf{z}_1, \ldots, \mathbf{z}_k), & \text{if } 1 \le i \le j \le m \\ -1, & \text{if } i = j + 1 \\ 0, & \text{if } i > j + 1 \end{cases}$$

- Charlie learns $C = F'(\mathbf{x}_1, \ldots, \mathbf{x}_k)$: For each $1 \le i \le j \le m$, Charlie learns $C_{i,j} = F'_{i,j}(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ via a PSM scheme (Figure 3) for $\mathbf{AFFINE}^k$.
- Charlie outputs $\det C$

Fig. 4. The $(\frac{1}{2}m(m+1)\log|\mathcal{R}|)$-bit PSM protocol for $\mathbf{BP}^k_m$.

Theorem A.4. *There is a PSM scheme (Figure 4) for $\mathbf{BP}^k_m$ over commutative ring $\mathcal{R}$ such that every party sends $\frac{m(m+1)}{2}$ ring elements.*

Proof. Correctness is straight-forward, as

$$\det C = \det F'(\mathbf{x}_1, \ldots, \mathbf{x}_k) = \det L \cdot \det F(\mathbf{x}_1, \ldots, \mathbf{x}_k) \cdot \det R = \det F(\mathbf{x}_1, \ldots, \mathbf{x}_k).$$

Privacy follows from the following observations:

- The distribution of $C = L \cdot F(\mathbf{x}_1, \ldots, \mathbf{x}_k) \cdot R$ is determined by $\det F(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ (Lemma A.3);
- For each $1 \le i \le j \le m$, Charlie learns $C_{i,j}$ in an independent nested PSM scheme, the corresponding messages can be simulated given C.

Thus Charlie's view can be simulated by first sampling C given $\det F(\mathbf{x}_1, \ldots, \mathbf{x}_k)$, then simulating the messages using C and the simulator of the underlaying PSM. □