Weakly Private Distributed Multi-User Secret Sharing

Joy Z. Wan Ning Luo

***Abstract*—Distributed multi-user secret sharing is a cryptographic primitive for secure distributed computing. Khalesi et al. (2021) provided a full information-theoretic characterization of the capacity region under weak privacy and correctness constraints. Their encoding and decoding scheme relied on a randomized Monte-Carlo procedure whose success probability depended on the choice of a very large finite field. A gap remains between their information-theoretic characterization and deterministic design which eliminates random failures and enables reproducible, hardware-efficient implementations. We close this gap by developing a deterministic, polynomial-time algorithm that achieves every integral-rate point in the *same* capacity region over small hardware-friendly fields. The algorithm is asymptotically faster than a single Monte-Carlo run. This work provides the first explicit deterministic realization of the previously known capacity region, advancing the algorithmic foundations of distributed secret sharing.**

***Index Terms*—Secret sharing, distributed storage, multi-user secrecy, capacity region, polynomial evaluation and interpolation.**

I. Introduction

A weakly private distributed system for multi-user secret sharing [18] [13] is specified by a *connected* bipartite graph $G = (U, V; E)$ where $U = \{0, 1, \cdots, |U| - 1\}$ is the set of at least two users, V is the set of at least two equal-size storage nodes, and E is the set of connections between users and storage nodes (see Figure 1). A dealer has direct access to all storage nodes, and holds $|U|$ independent secret messages of various sizes, one for each user. The dealer encodes the secret messages and additional random noise messages jointly into $|V|$ shares of the same size as a storage node, and places the encoded shares at the storage nodes respectively. The encoding scheme must meet two requirements:

1) *correctness*: each user can decode its own message from the shares at its neighboring storage nodes,
2) *weak privacy*: no user can learn, in an information theoretic sense [17], any information about the message of any other user from the shares at its neighboring storage nodes.

Such distributed system extends the classic Shamir's secret sharing scheme [16] broadly by supporting simultaneously *multiple* secret messages of *various* sizes and allowing *arbitrary* access structures. By defining the *rate* of a secret message as the size of the message normalized by the size of a storage node, the *capacity region* of the system is the set of all achievable rate tuples indexed by users.

Siebel School of Computing and Data Science, University of Illinois Urbana-Champaign

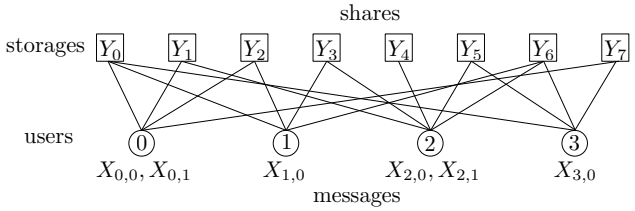

Fig. 1. Bipartite graph representation of a distributed system for multi-user secret sharing.

The distributed system for multi-user secret sharing was first proposed by Soleymani and Mahdavifar [18] in the *regular* setting, and then studied by Khalesi et al. [13] in general setting (cf . [18] and [13] for more background and applications). In contrast to the *locality* of decoding, the encoding of secret and noise messages into shares is *global*. This globality of encoding is illustrated by the running example of the system shown in Figure 1. In this example, the dealer has 2 secret symbols $X_{0,0}$ and $X_{0,1}$ for user 0, 1 secret symbol $X_{1,0}$ for user 1, 2 secret symbols $X_{2,0}$ and $X_{2,1}$ for user 2, and 1 secret symbol $X_{3,0}$ for user 3. Each secret symbol is a member of a finite field $\mathbb{F}$ with $|\mathbb{F}| = 7$. From the 6 secret symbols and 2 additional noise symbols R_1 and R_2 in $\mathbb{F}$, the dealer generates 8 shares according to the scheme:

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 3 & 3 & 2 & -1 & -1 & -2 \\ -2 & -1 & 3 & 2 & -2 & 2 & 1 & -2 \\ 1 & -2 & 3 & -2 & 3 & 0 & 2 & -2 \\ -3 & -3 & 2 & -3 & 0 & -3 & 0 & -2 \\ 1 & -2 & -1 & 3 & -2 & -3 & 0 & -2 \\ -1 & 1 & -3 & 0 & -3 & 3 & -2 & -2 \\ 0 & 3 & -2 & -2 & -3 & 1 & -2 & -2 \\ -2 & 0 & 3 & 1 & 1 & -2 & 3 & -2 \end{bmatrix} \begin{bmatrix} X_{0,0} \\ X_{0,1} \\ R_1 \\ X_{1,0} \\ R_2 \\ X_{2,0} \\ X_{2,1} \\ X_{3,0} \end{bmatrix}. \quad (1)$$

The decoding scheme is local:

$$\begin{bmatrix} X_{0,0} \\ X_{0,1} \\ X_{1,0} \\ X_{2,0} \\ X_{2,1} \\ X_{3,0} \end{bmatrix} = \begin{bmatrix} 1 & 2 & -1 & 0 & 0 & 0 & 0 & -2 \\ 2 & 2 & 2 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 2 & 2 & 0 & 0 & 1 & 0 \\ 0 & 3 & 0 & -1 & 0 & -3 & 1 & 0 \\ 0 & -3 & 0 & -1 & -1 & 1 & -3 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 & 2 & -2 \end{bmatrix} \begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{bmatrix}.$$

However, the share Y_1 depends on *all* secret/noise symbols; and the share Y_4 at the storage node with only one neighbor depends on *all but one* secret/noise symbols.

Technically, the general distributed system follows the same mathematical principle of multipoint polynomial interpolation over a finite field as Shamir's secret sharing scheme [16], but is much more challenging.

- Polynomials at different users may have different degrees. Such *degree diversity* imposes an algorithmic challenge for fast polynomial interpolation (and ultimately, fast Fourier transform) over a finite field (cf. [4]).
- Each share must be scaled by a *non-zero* factor so that the *scaled share* is the value for interpolation. The scaling factors may vary with both storage nodes and users. The proper selection of the $|E|$ scaling factors is a major task in the design of encoding/decoding schemes.
- Unlike Shamir's secret sharing scheme [16], the shares is generated consistently from $|U|$ polynomials at $|E|$ points in total.

Khalesi et al. [13] remarkably discovered an explicit form of the capacity region. For each $S \subseteq U$, $N(S)$ denotes the set of storage nodes neighboring to S; and if S is singleton $\{u\}$, then $N(\{u\})$ is simply written as $N(u)$. Let $\Delta := \max_{u\in U} |N(u)| \geq 2$ be the *maximum degree* of all users in G, and b be the "*private-degree*" vector indexed by U defined by

$$b_u := \min_{u'\in U\setminus\{u\}} |N(u) \setminus N(u')|, \ \forall u \in U. \tag{2}$$

Assume that each message symbol is independently and uniformly distributed on a finite field $\mathbb{F}$ with $|\mathbb{F}| > \Delta$. Then the capacity region consists of all nonnegative rate vectors r indexed by U satisfying

- private-degree constraint: $r \leq b$;
- sharing constraint: $r(S) := \sum_{u\in S} r_u \leq |N(S)|, \ \forall S \subseteq U$.

The set of nonnegative r meeting the sharing constraint only is known as a *transversal polymatroid* [11] or a *Boolean polymatroid* [10]. Transversal and Boolean polymatroids are classical structures in combinatorial optimization that characterize feasible capacity allocations under submodular constraints. Thus the capacity region is the upper truncation of a transversal polymatroid by the integer vector b, and hence all its vertices are integer-valued (cf. [7], [9]).

While the outer bound direction of the capacity region was relatively easy (cf. Section V in [13]), the inner bound direction was very involved (cf. Section IV in [13]). Khalesi et al. [13] asserted the existence of a weakly private encoding/decoding scheme for secret messages of *integral* rates in the capacity region; and an arbitrary member in the capacity region is then achievable through spatial or temporal sharing based on Carathéodory's theorem. They also gave a randomized Monte-Carlo algorithm based on the Schwartz–Zippel lemma [15], [20] for constructing the encoder/decoder. However, their randomized Monte-Carlo algorithm is quite computationally expensive:

- A *single* Monte-Carlo run has $O(|U|^3 |V|^3)$ time complexity.
- To ensure success with high probability, the finite field size $|\mathbb{F}|$ must be sufficiently larger than $|E|$ rather than just Δ.
- There is no polynomial guarantee on the number of trials required for success.

We remark that when counting the running time of an algorithm, each arithmetic operation over the ambient field has a unit computational time conventionally.

In this paper, we present a fully *deterministic* algorithm for encoding/decoding secret messages of *integral* rates r in the known capacity region. Unlike earlier randomized schemes, a deterministic design ensures reproducibility and removes the possibility of encoding failures due to unfavorable random seeds, while allowing operation over much smaller finite fields. Our deterministic algorithm has worst-case time complexity

$$O\left(\min\{|U|, r(U)^{1/2}\}\,|E| + |V|^{\omega}\right), \tag{3}$$

where $\omega > 2$ is the best-known upper bound (about 2.371339 in [2]) on the fast matrix multiplication exponent. Since $r(U) \leq |V|$ and $|E| \leq |U|\,|V|$, the time complexity on dense graph G is bounded by

$$O\left(\min\{|U|, |V|^{1/2}\}\ |U|\,|V| + |V|^{\omega}\right).$$

Asymptotically, our deterministic algorithm is orders of magnitude faster than a single Monte-Carlo run of Khalesi et al. [13]. Our algorithm also guarantees correctness whenever $|\mathbb{F}| > \Delta$. This significantly relaxes the field-size requirement and makes the scheme feasible over small hardware-friendly fields. Moreover, the resulting encoder achieves $O(|V|^2)$-time global encoding complexity and each user u performs local decoding in $O(|N(u)| \log^2 |N(u)|)$ time.

The superior performance of our deterministic algorithm is mainly due to two new algorithmic primitives in graph theory and algebraic computation. In fact, in the complexity bound given in equation (3), the first term is the time complexity for computing a perfect r-star matching defined later in Section II, and the second term is the time complexity of inverting a $|V|\times|V|$ matrix with diagonal indeterminates. Both algorithmic primitives are of independent interest. Our algorithm also leverages recent advances in multipoint polynomial evaluation and interpolation over finite fields (cf. [4]). Suppose $1 \leq n \leq |\mathbb{F}|$. Each vector $a = [a_0, a_1, \cdots, a_{n-1}]^\top \in \mathbb{F}^n$ defines a univariate polynomial

$$p_a(x) := \textstyle\sum_{i=0}^{n-1} a_i x^i. \tag{4}$$

- **Multipoint Evaluation**: Given $a \in \mathbb{F}^n$ and any m evaluation points $x_0, \cdots, x_{m-1}$ in $\mathbb{F}$, $p_a(x_i)$ for $i = 0, 1, \cdots, m-1$ can be computed in $O(n \log^2 n + m \log^2 m)$ time (cf. Theorem 7.2 in [4]).
- **Multipoint Interpolation**: Given n *distinct* interpolation points $x_0, \cdots, x_{n-1}$ in $\mathbb{F}$ and n value points $y_0, \cdots, y_{n-1}$ in $\mathbb{F}$, the unique $a \in \mathbb{F}^n$ such that $p_a(x_i) = y_i$ holds for $i = 0, 1, \cdots, n-1$ can be computed in $O(n \log^2 n)$ time (cf. Theorem 7.3 in [4]).

The remaining of this paper is organized as follows. In Section II, we characterize the integral members r of the capacity region in terms of perfect r-star matchings and develop an efficient algorithm for constructing a perfect r-star matching. In Section III, we give deterministic algorithms for inverting a diagonally parametric matrix. In Section IV, we devise a weakly private encoding/decoding scheme for secret messages with integral rates in the capacity region. Finally, we conclude this paper in Section V.

Notations: For any two integers $n_1 \le n_2$, $[n_1 : n_2]$ denotes the set $\{n_1, n_1+1, \cdots, n_2\}$. $\mathbb{R}_+$ (resp., $\mathbb{Z}_+$) stands for the set of nonnegative reals (resp., integers). $\mathbb{R}_+^U$ (resp., $\mathbb{Z}_+^U$) is the set of non-negative real (resp., integer) vectors indexed by users in U. For each $x \in \mathbb{R}_+^U$ and each $S \subseteq U$, $x(S)$ is the sum $\sum_{u\in S} x_u$. $\mathbf{I}_n$ represents the $n \times n$ identity matrix over $\mathbb{F}$, and the subscript may be omitted if it is clear from the context. Each vector $a = [a_0, a_1, \cdots, a_{n-1}]^\top \in \mathbb{F}^n$ defines a diagonal matrix

$$diag(a) := \begin{bmatrix} a_0 & & & \\ & a_1 & & \\ & & \ddots & \\ & & & a_{n-1} \end{bmatrix}.$$

II. Perfect r-star matchings

The capacity region of $G = (U, V; E)$ given by Khalesi et al. [13] is neat and clean. However, the sharing constraint on exponentially many sets is less transparent and convenient for membership certification. We introduce the concept of star matching to facilitate simple membership certification of integral rates and enable easy applications for encoding/decoding design. A set of edges $M \subseteq E$ is called a *star matching* if each node in V is incident to at most one edge in M. Equivalently, M is a star matching if and only if M can be partitioned into node-disjoint stars with centers in U. For any $r \in \mathbb{Z}_+^U$, an *r-star matching* is a star matching M in which each node u is incident to *at most* r_u edges in M. An r-star matching is *perfect* if each node u is incident to *exactly* r_u edges in M (see an example in Figure 2). A perfect r-star matching ensures that every subset $S \subseteq U$ has at least $r(S)$ neighbors, hence certifies that r satisfies the sharing constraint. Theorem 2.2 asserts that the opposite also holds. Thus, the membership certification of r amounts to finding a perfect r-star matching if there is any. Throughout this section we fix an $r \in \mathbb{Z}_+^U$ with $r \le b$, where b is the private-degree vector defined in equation (2).

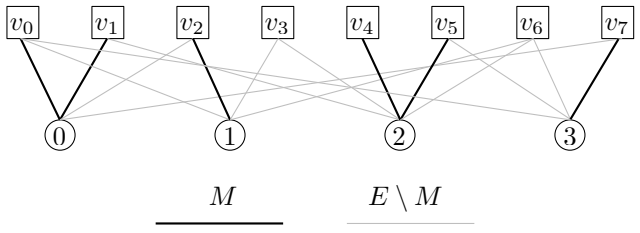

Fig. 2. A perfect r-star matching consisting of blue edges for $r = [2, 1, 2, 1]^\top$.

We study perfect r-star matchings from the broader context of maximum r-star matchings. An r-star matching with the maximum size is called a *maximum* r-star matching. A perfect r-star matching is a maximum r-star matching, but not vice versa. Let φ be the maximum size of r-star matchings. It is trivial that $\varphi \le \min\{|V|, r(U)\}$, and $\varphi = r(U)$ if and only if there is a perfect r-star matching. We shall derive the explicit expression of φ, and present an algorithm for finding a maximum r-star matching in $O(\min\{|U|, \varphi^{1/2}\}|E|)$ time. The expression of φ would be utilized to characterize the existence of perfect r-star matchings. The algorithm combines ideas from two alternative approaches described below, which *fail* to achieve the targeted time complexity individually.

Computing a maximum r-star matching can be reduced to an instance of computing a maximum bipartite matching defined as follows. Let G' be the graph obtained from G by splitting each vertex $u \in U$ into r_u copies and replacing each edge uv by r_u edges connecting the r_u copies of u with v (see Figure 3). The graph G' has $\sum_{u\in U} r_u |N(u)|$ edges. A maximum bipartite matching in G' can be computed by the Hopcroft-Karp algorithm [12] (see also Theorem 16.5 in [14], for instance) in $O(\varphi^{1/2} \sum_{u\in U} r_u |N(u)|)$ time, which exceeds the targeted $O(\varphi^{1/2}|E|)$ complexity. This reduction was adopted in [13], which incurs an $O(|V|^{2.5})$ computation cost for any r satisfying the sharing constraint.

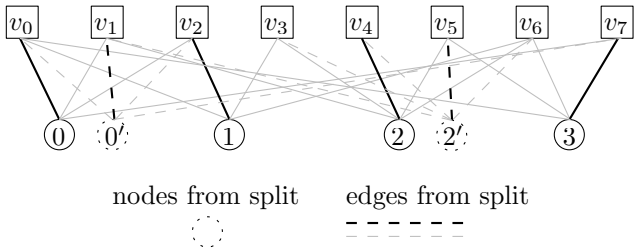

Fig. 3. Reduction to maximum bipartite matching by node splitting.

Computing a maximum r-star matching can also be reduced to an instance of computing maximum edge-disjoint paths defined as follows. Let G'' be the digraph (see Figure 4) obtained from G by

- orienting all edges in E from U to V,
- adding a source node s and connecting s to each $u \in U$ with r_u parallel edges, and
- adding a sink node t and an edge (v, t) for each $v \in V$.

Then each r-star matching in G corresponds to a collection of edge-disjoint s-t paths in G'' of the same size, and vice versa. The graph G'' has at most $2|E| + |V|$ edges. Hence a maximum collection of edge-disjoint s-t paths in G'' can be computed in $O(|E|^{3/2})$ time by the Even-Tarjan algorithm [8] (see also Corollary 9.6a in [14], for instance), which again exceeds the targeted $O(\varphi^{1/2}|E|)$ complexity.

Instead, we take a folklore reduction of (maximum) r-star matchings to (maximum) *integral* flows. Let D be the flow network (see Figure 5) obtained from G by

- orienting all edges in E from U to V each having *infinite* capacity (or a finite capacity $2|V|$ so that none of these edges gets saturated by any feasible flow),

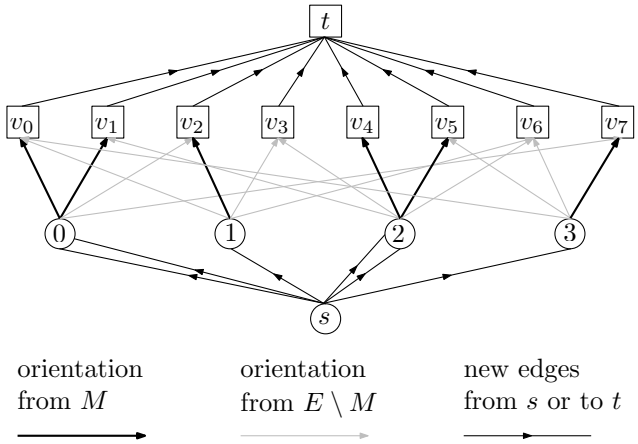

Fig. 4. Reduction to maximum edge-disjoint paths by edge replication.

- adding a source node s and an edge (s, u) of capacity r_u for each $u \in U$,
- adding a sink node t and an edge (v, t) of *unit* capacity for each $v \in V$.

For any integral s-t flow f in D, $val(f)$ denotes the value of f, and M_f denotes the set of edges in E whose orientations in D carry a unit flow of f. Then M_f is an r-star matching, and $\left|M_f\right| = val(f)$. Conversely, for any r-star matching M, there exists an integral s-t flow f in D such that $M = M_f$. By the integral flow theorem (see Corollary 10.3a in [14], for instance), finding a maximum r-star matching reduces to find a maximum integral s-t flow in D, and φ is the maximum s-t flow value in D. Using the max-flow min-cut duality (see Theorem 10.3 in [14] or Theorem 24.6 in [5], for instance), we derive the explicit expression of φ below.

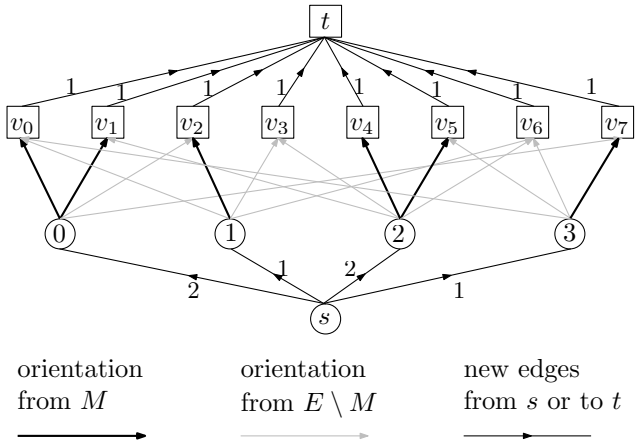

Fig. 5. The flow network D constructed from G.

Lemma 2.1: $\varphi = r(U) + \min_{S \subseteq U} [|N(S)| - r(S)]$.

Proof. For any $S \subseteq U$, $\{s\} \cup S \cup N(S)$ is an s-t cut with capacity

$$|N(S)| + r(U \setminus S) = r(U) + [|N(S)| - r(S)] .$$

Hence, the minimum s-t cut capacity is at most φ. It remains to show that for some $S \subseteq U$, $\{s\} \cup S \cup N(S)$ is a minimum s-t cut. Consider any minimum s-t cut C, and let $S = C \cap U$. As C has finite capacity, it is necessary that $N(S) \subseteq C \cap V$. Furthermore, the equality must hold, for otherwise deleting any node $v \in (C \cap V) \setminus N(S)$ from C would yield a cut with strictly smaller capacity than C. Hence, $\{s\} \cup S \cup N(S)$ is the minimum s-t cut C. So, the minimum s-t cut capacity is exactly φ. The lemma then follows from the max-flow min-cut duality. ■

From Lemma 2.1, we are now able to characterize the existence of perfect r-star matching in terms of sharing constraint, or a source-saturating flow in D (a flow saturating all edges leaving the source). Hence, the existence of a perfect r-star matching certifies that r satisfies the sharing constraint.

Theorem 2.2: The following statements are equivalent:

1) There is a perfect r-star matching.
2) D has a source-saturating flow.
3) $r(S) \leq |N(S)|$ for each each $S \subseteq U$.

Proof. By definition, there is a perfect r-star matching if and only if $\varphi = r(U)$. As φ is also the maximum s-t flow value in D, $\varphi = r(U)$ is equivalent to that D has a source-saturating flow. By Lemma 2.1, $\varphi = r(U)$ is also equivalent to

$$\min_{S \subseteq U} [|N(S)| - r(S)] = 0.$$

Note that $|N(\emptyset)| - r(\emptyset) = 0$. The above equality holds exactly when $r(S) \leq |N(S)|$ for each $S \subseteq U$. Thus, the theorem holds. ■

We continue to compute a maximum s-t flow in D, from which we can get a maximum r-star matching by our reduction. We restore D to the "slim" variant in which all edges oriented from E have *unit* capacity, so that the residual graph of any feasible flow is much simpler. Such modification preserves the set of feasible s-t flows, because each feasible flow carries at most one unit of flow on those altered edges due to the flow-conservation constraints at V and the unit-capacity of all edges leaving V. Our algorithm is a **specialization** of the Dinitz's blocking flow method [6] for max-flow (see also Section 10.6 in [14], for instance) on the slim variant of D. Just as the two earlier alternative reductions, the targeted time complexity cannot be achieved by applying the original Dinitz's algorithm as a blackbox. Instead, we have to open the box and provide a specialized implementation of the generic blocking flow method using the ideas from the two earlier alternative reductions.

We give a high-level overview of the two key concepts of level graph and blocking flow with respect to an integral s-t flow f. Let D_f be the residual graph of f, and c_f be the residual edge capacity function. The *level graph* L_f is a subgraph of D_f which retains only those edges contained in at least one shortest s-t path in D_f. It is acyclic and can be constructed easily by *breadth*-first search in $O(|E|)$ time. The

s-t paths in L_f are precisely the shortest s-t paths in D_f. An s-t flow f' in L_f is *blocking* if after deleting all edges in L_f saturated by f', the resulting subgraph of L_f contains no s-t path. An iteration of Dinitz's method finds a blocking f' in L_f, and augments f by f'. After each blocking flow augmentation, the s-t distance (in terms of shortest path length) in D_f *strictly increases*.

Dinitz's method starts with the 0-flow f, and repeat the following iteration of blocking flow augmentation until there is no s-t path in D_f:

- Construct L_f from D_f.
- Compute a blocking flow f' in L_f.
- Augment f by f'.

The first operation and the third operation takes $O(|E|)$ time. The second operation is reduced to computing an inclusion-wise *maximal* collection of edge-disjoint paths in a directed acyclic graph as follows.

- Construct L_f^+ from L_f by replacing each edge (s,u) with $c_f(u) = r(u) - f(u)$ parallel edges. Then L_f^+ is still acyclic and has at most $2|E| + |V|$ edges.
- Compute a maximal collection of edge-disjoint s-t paths in L_f^+, which yields an integral blocking s-t flow in L_f.

The first step takes $O(|E|)$ time. Since L_f^+ is acyclic and has $O(|E|)$ edges, the second step can be implemented in $O(|E|)$ using *depth*-first search (see Theorem 9.4 in [14], for instance). Thus, each iteration can be implemented in $O(|E|)$ time.

It remains to show that there are $O(\min\{|U|, \varphi^{1/2}\})$ iterations of blocking flow augmentation. Underlying this upper bound is the property that the s-t distance in D_f strictly increases after each iteration. Note that the s-t distance in D_f is always an *odd* number at least 3. Thus, after k iterations, the s-t distance in D_f is an odd number at least $2k+3$. As the s-t distance in D_f is no more than $2(|U|-1)+3$, the number of iterations is no more than $|U| - 1 = O(|U|)$. The other upper bound $O(\varphi^{1/2})$ on the number of iterations is established through the progress of the flow f towards a maximum s-t flow measured by the optimality gap $\varphi - val(f) = \varphi - |M_f|$.

The lemma below asserts that the optimality gap drops at least *harmonically* with the number of iterations.

Lemma 2.3: After k iterations of blocking flow augmentation, $\varphi - |M_f| \le \frac{\varphi}{k+1}$

Proof. Let f^* be an integral maximum s-t flow so that $val(f^*) = |M_{f^*}| = \varphi$. We define an integral s-t flow g in D_f as follows.

- For each edge e in D with $f(e) < f^*(e)$, e is an edge in D_f, and we set $g(e) = f^*(e) - f(e)$.
- For each edge e in D with $f(e) > f^*(e)$, its reverse e^{-1} is an edge in D_f, and we set $g(e^{-1}) = f(e) - f^*(e)$.
- On all other edges in D_f, g has value 0.

Then g is an integral s-t flow in D_f of value $val(f^*) - val(f) = \varphi - |M_f|$, and takes 0/1 values on the edges *not* incident to s or t.

By conformal flow decomposition (see Theorem 11.1 in [14], for instance), there are $\varphi - |M_f|$ s-t paths P_i in D_f for $i = 1, \cdots, \varphi - |M_f|$ such that the sum of unit path-flows along these paths is at most g. For each $1 \le i \le \varphi - |M_f|$, let P_i' be the path obtained from P_i by removing the first edge and the last edge. Then each P_i' has length at least $(2k+3) - 2 = 2k+1$; hence the total length of P_i' for $1 \le i \le \varphi - |M_f|$ is at least $(2k+1)(\varphi - |M_f|)$. On the other hand, the paths P_i' for $1 \le i \le \varphi - |M_f|$ are edge-disjoint, and the undirected version of each P_i' alternates between M_{f^*} and M_f. Hence the total length of the paths P_i' for $1 \le i \le \varphi - |M_f|$ is at most $|M_{f^*}| + |M_f| = \varphi + |M_f|$. Thus,

$$(2k+1)(\varphi - |M_f|) \le \varphi + |M_f|$$

implying $|M_f| \ge \varphi \frac{k}{k+1}$. Consequently,

$$\varphi - |M_f| \le \frac{\varphi}{k+1};$$

and the lemma follows. ■

From Lemma 2.3, the upper bound $O(\varphi^{1/2})$ on the number of iterations easily follows.

Lemma 2.4: There are at most $2\lfloor \varphi^{1/2} \rfloor$ blocking flow augmentations.

Proof. After $\lfloor \varphi^{1/2} \rfloor$ blocking flow augmentations, the optimality gap is at most

$$\frac{\varphi}{\lfloor \varphi^{1/2} \rfloor + 1} \le \varphi^{1/2},$$

hence is at most $\lfloor \varphi^{1/2} \rfloor$. Since each subsequent augmentation reduces the optimality gap by at least one, at most $\lfloor \varphi^{1/2} \rfloor$ more augmentations are needed. ■

Hence, our specialization of Dinitz's blocking flow method computes a maximum r-star matching in

$$O(\min\{|U|, \varphi^{1/2}\}|E|)$$

time. If r satisfies the sharing constraint, then $\varphi = r(U) = \sum_{u \in U} r_u$ and the algorithm computes a perfect r-star matching in $O(\min\{|U|, r(U)^{1/2}\}|E|)$ time.

Therefore, for any integral r satisfying the private-degree constraint, the existence of a perfect r-star matching exactly characterizes the membership of r in the integral capacity region, and such matchings can be computed in near-optimal time.

III. Inverting a diagonally parametric matrix

Suppose $\mathbb{F}$ is a finite field with $|\mathbb{F}| \ge 3$. Let $\lambda = [\lambda_0, \lambda_1, \cdots, \lambda_{n-1}]^\top$ be a vector of $n \ge 1$ *variables* (or *indeterminates*). For any fixed matrix $\mathbf{C} = [c_{i,j}]_{i,j=0}^{n-1} \in \mathbb{F}^{n \times n}$ without variables, the matrix $\mathbf{A} := diag(\lambda) + \mathbf{C}$ is called a

diagonally parametric matrix over $\mathbb{F}$. These diagonally parametric matrices constitute the algebraic core of the encoder transformation in Section IV, where the diagonal parameters in λ represent deterministic *scaling factors* of the storage nodes. It is imperative for the encoder to choose $\lambda \subseteq (\mathbb{F} \setminus \{0\})^n$ ensuring invertibility of $\mathbf{A}$. Note that $\det(\mathbf{A})$ is a polynomial in $\lambda_0, \lambda_1, \cdots, \lambda_{n-1}$ over $\mathbb{F}$ of total degree n and the coefficient of the monomial $\lambda_0\lambda_1\cdots\lambda_{n-1}$ in $\det(\mathbf{A})$ is 1. By Combinatorial Nullstellensatz (cf. Theorem 1.2 in [3]) and the condition that $|\mathbb{F}| \geq 3$, $\mathbf{A}$ is invertible at some $\lambda \in \{\pm 1\}^n \subseteq \mathbb{F}^n$. Henceforth, inverting a diagonally parametric matrix $\mathbf{A}$ means seeking both $\lambda \subseteq \{\pm 1\}^n$ ensuring the invertibility of $\mathbf{A}$ and the inverse of $\mathbf{A}$ at such λ.

In this section, we will give two algorithms for inverting diagonally parametric matrices.

- The incremental inversion algorithm is conceptually simple and has cubic time complexity. It is suitable for small applications.
- The divide-and-conquer inversion algorithm is asymptotically faster with $O(n^\omega)$ complexity, where $\omega \geq 2$ is the fast matrix multiplication exponent.

These two algorithms will be presented in the two subsections respectively. In the sequel we give a small example and an overview of blockwise matrix inversion.

Consider an instance with $|\mathbb{F}| = 7$ and

$$\mathbf{A} = \begin{bmatrix} \lambda_0 & 0 & 3 & 3 \\ 0 & \lambda_1 & -3 & 2 \\ 0 & 0 & \lambda_2 & -1 \\ 0 & -3 & -1 & \lambda_3 + 3 \end{bmatrix}.$$

For $\lambda_0 = \lambda_1 = \lambda_2 = \lambda_3 = 1$, $\mathbf{A}$ is singular. But for $\lambda_0 = \lambda_1 = \lambda_2 = 1$ and $\lambda_3 = -1$, $\mathbf{A}$ has determinant -2 and the corresponding inverse

$$\mathbf{A}^{-1} = \begin{bmatrix} 1 & 2 & -1 & 3 \\ 0 & 3 & -2 & 3 \\ 0 & 2 & 3 & 3 \\ 0 & 2 & 2 & 3 \end{bmatrix}.$$

The inverting algorithms are built upon blockwise matrix inversion. Suppose a square matrix $\mathbf{M}$ over a finite field $\mathbb{F}$ is partitioned into blocks

$$\begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix}$$

in which $\mathbf{M}_{11}$ is a square *invertible* matrix. The *Schur complement* of the block $\mathbf{M}_{11}$ in $\mathbf{M}$ is

$$\mathbf{M}/\mathbf{M}_{11} := \mathbf{M}_{22} - \mathbf{M}_{21}\mathbf{M}_{11}^{-1}\mathbf{M}_{12}.$$

The Block LU Decomposition of $\mathbf{M}$ is given by

$$\mathbf{M} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{M}_{21}\mathbf{M}_{11}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{0} & \mathbf{M}/\mathbf{M}_{11} \end{bmatrix},$$

and the Block LDU Decomposition of $\mathbf{M}$ is given by

$$\mathbf{M} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{M}_{21}\mathbf{M}_{11}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{M}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}/\mathbf{M}_{11} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{M}_{11}^{-1}\mathbf{M}_{12} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

Both decompositions result from blockwise Gaussian eliminations on the matrix $\mathbf{M}$. The Block LDU Decomposition of $\mathbf{M}$ implies that

$$\det(\mathbf{M}) = \det(\mathbf{M}_{11}) \det(\mathbf{M}/\mathbf{M}_{11});$$

hence $\mathbf{M}$ is invertible if and only if $\mathbf{M}/\mathbf{M}_{11}$is invertible. In addition, if $\mathbf{M}/\mathbf{M}_{11}$is invertible, then $\mathbf{M}^{-1}$ is

$$\begin{bmatrix} \mathbf{M}_{11}^{-1} + \mathbf{M}_{11}^{-1}\mathbf{M}_{12}\left(\mathbf{M}/\mathbf{M}_{11}\right)^{-1}\mathbf{M}_{21}\mathbf{M}_{11}^{-1} & -\mathbf{M}_{11}^{-1}\mathbf{M}_{12}\left(\mathbf{M}/\mathbf{M}_{11}\right)^{-1} \\ -\left(\mathbf{M}/\mathbf{M}_{11}\right)^{-1}\mathbf{M}_{21}\mathbf{M}_{11}^{-1} & \left(\mathbf{M}/\mathbf{M}_{11}\right)^{-1} \end{bmatrix}.$$

A. Incremental inversion

For each $i \in [0 : n-1]$, let $\mathbf{A}_i$ be the i-th leading principal submatrix of $\mathbf{A}$ consisting of all entries in the rows $[0 : i]$ and columns $[0 : i]$. For $i = 0, 1, \cdots, n-1$ *successively*, the iteration fixes λ_i to either 1 or -1 such that $\mathbf{A}_i$ is invertible, and at the same time computes $\mathbf{A}_i^{-1}$. We elaborate on each iteration below.

In the iteration $i = 0$, the matrix $\mathbf{A} = \begin{bmatrix} \lambda_0 + c_{0,0} \end{bmatrix}$ is singular exactly when $\lambda_0 = -c_{0,0}$. As the set $\{\pm 1\} \setminus \{-c_{0,0}\}$ is non-empty, assigning any member from this set to λ_0 makes $\mathbf{A}$ invertible. Thus, we choose

$$\lambda_0 = \begin{cases} -1, & \text{if } c_{0,0} = -1; \\ 1, & \text{else} \end{cases}$$

and compute $\mathbf{A}_0^{-1} = [(\lambda_0 + c_{0,0})^{-1}]$. Both λ_0 and $\mathbf{A}_0^{-1}$ can be computed in $O(1)$ time.

Now consider each iteration $i \in [1 : n-1]$. At the beginning of the iteration i, we have already selected $\lambda_0, \cdots, \lambda_{i-1}$ and derived $\mathbf{A}_{i-1}^{-1}$. We partition $\mathbf{A}_i$ into four blocks as

$$\mathbf{A}_i = \begin{bmatrix} \mathbf{A}_{i-1} & v \\ u^\top & \lambda_i + c_{i,i} \end{bmatrix}.$$

As $\mathbf{A}_{i-1}$ is invertible, the Schur complement of the block $\mathbf{A}_{i-1}$ in $\mathbf{A}_i$ is

$$\mathbf{A}_i/\mathbf{A}_{i-1} = \lambda_i + c_{i,i} - u^\top \mathbf{A}_{i-1}^{-1} v.$$

Thus, $\mathbf{A}_i$ is singular exactly when $\lambda_i = u^\top \mathbf{A}_{i-1}^{-1} v - c_{i,i}$. As the set $\{\pm 1\} \setminus \left\{u^\top \mathbf{A}_{i-1}^{-1} v - c_{i,i}\right\}$ is non-empty, assigning any member from this set to λ_i makes $\mathbf{A}_i$ invertible. Thus, we choose

$$\lambda_i = \begin{cases} -1, & \text{if } u^\top \mathbf{A}_{i-1}^{-1} v - c_{i,i} = 1; \\ 1, & \text{else} \end{cases}$$

and compute

$$\theta = \left(\lambda_i + c_{i,i} - u^\top \mathbf{A}_{i-1}^{-1} v\right)^{-1},$$

$$\mathbf{A}_i^{-1} = \begin{bmatrix} \mathbf{A}_{i-1}^{-1} + \theta \mathbf{A}_{i-1}^{-1} v u^\top \mathbf{A}_{i-1}^{-1} & -\theta \mathbf{A}_{i-1}^{-1} v \\ -\theta u^\top \mathbf{A}_{i-1}^{-1} & \theta \end{bmatrix}.$$

Both λ_i and $\mathbf{A}_i^{-1}$can be computed in three steps.

- Step 1: Compute

$$\mathbf{A}_{i-1}^{-1} v, u^\top \mathbf{A}_{i-1}^{-1}, u^\top \mathbf{A}_{i-1}^{-1} v, \mathbf{A}_{i-1}^{-1} v u^\top \mathbf{A}_{i-1}^{-1}$$

  in $O(i^2)$ time.

- Step 2: Compute λ_i and θ in $O(1)$ time.
- Step 3: Compute $\mathbf{A}_i^{-1}$ in $O((i+1)^2)$ time.

Thus iteration i takes $O((i+1)^2)$ time, regardless of the sparsity of $\mathbf{A}_{i-1}$ or $\mathbf{C}$.

At the end of the iteration $n-1$, all λ_i for $i \in [0:n-1]$ and $\mathbf{A}^{-1} = \mathbf{A}_{n-1}^{-1}$ are derived. The overall time complexity is $O(n^3)$.

As a running example, consider an instance with $|\mathbb{F}| = 7$ and

$$\mathbf{A} = \begin{bmatrix} \lambda_0 & 0 & 3 & 0 & 0 & 0 & 0 & 3 \\ 0 & \lambda_1 & -2 & 0 & 0 & 0 & 0 & 1 \\ 3 & 0 & \lambda_2 & 0 & 0 & 0 & 3 & 0 \\ -2 & 0 & 0 & \lambda_3 & 0 & 0 & 1 & 0 \\ 0 & 3 & 0 & 3 & \lambda_4 & 0 & 0 & 0 \\ 0 & -2 & 0 & 1 & 0 & \lambda_5 & 0 & 0 \\ 0 & -3 & 0 & 2 & 0 & 0 & \lambda_6 & 0 \\ 1 & 0 & 0 & 0 & 0 & -3 & 1 & \lambda_7 \end{bmatrix}. \quad (5)$$

This matrix arises in our running example of encoding in Section IV. The application of the incremental algorithm chooses $\lambda_i = 1$ for $0 \le i \le 6$ and $\lambda_7 = -1$, with the determinants of $\mathbf{A}_i$ for $1 \le i \le 7$ equal to

$$1, 1, -1, -1, -1, -1, -1, 2$$

respectively. The corresponding inverse of $\mathbf{A}$ is

$$\mathbf{A}^{-1} = \begin{bmatrix} 2 & -3 & 2 & -2 & 0 & 2 & 0 & 3 \\ 3 & 1 & 0 & 3 & 0 & 2 & 1 & 3 \\ -3 & -2 & -1 & 2 & 0 & 2 & -2 & 3 \\ -2 & 2 & 3 & -2 & 0 & 2 & -3 & 3 \\ -3 & -2 & -2 & -3 & 1 & 2 & -1 & 3 \\ 1 & 0 & -3 & 1 & 0 & 3 & -2 & 3 \\ -1 & -1 & 1 & -1 & 0 & 2 & 3 & 3 \\ -2 & 3 & -2 & 1 & 0 & 2 & 2 & 3 \end{bmatrix}. \quad (6)$$

The theorem below summarizes the performance of the incremental inversion algorithm.

Theorem 3.1: The incremental inversion algorithm inverts a diagonally parametric $n \times n$ matrix $\mathbf{A}$ in $O(n^3)$ time.

B. Divide-and-conquer inversion

In order to beat the cubic complexity of inverting diagonally parametric matrices, we now present a Strassen-type matrix inversion [19] enabling asymptotically faster computation. For an arbitrary matrix, the Strassen-type matrix inversion may fail. However, the binary freedom of choosing the diagonal elements (the scaling factors of storage nodes in our encode application) turns to be an algorithmic *advantage* guaranteeing the success. The algorithm follows a divide-and-conquer (D&C) strategy and exploits fast matrix multiplication, hence is called D&C inversion. No assumption is made on matrix multiplication algorithm that is used, except that its complexity is $O(n^\omega)$ for some $\omega \ge 2$. The D&C inversion will also achieve asymptotic computational complexity $T(n) = O(n^\omega)$.

Assume that n is an exact power of 2; we'll see at the end of the subsection what to do if n is not an exact power of 2. The base setting is $n = 1 = 2^0$. Then the problem can be solved in $O(1)$ time as in the first iteration of the incremental algorithm. Thus, $T(1) = O(1)$.

Suppose $n > 1$. We partition $\mathbf{A}$ in four $\frac{n}{2} \times \frac{n}{2}$ blocks:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix},$$

and proceed in four steps.

- Step 1: Invert $\mathbf{A}_{11}$ recursively to get the ± 1-values of the first $\frac{n}{2}$ diagonal variables and $\mathbf{A}_{11}^{-1}$.
- Step 2: Compute the Schur complement

  $$\mathbf{A}/\mathbf{A}_{11} = \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12}.$$

  It is important to note that $\mathbf{A}_{21}$, $\mathbf{A}_{12}$, and now $\mathbf{A}_{11}^{-1}$ have *no* variables, hence $\mathbf{A}/\mathbf{A}_{11}$ is also a diagonally parametric matrix. We compute and store 3 matrix multiplications $\mathbf{A}_{21}\mathbf{A}_{11}^{-1}$, $\mathbf{A}_{11}^{-1}\mathbf{A}_{12}$, and $\mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12}$, and then create $\mathbf{A}/\mathbf{A}_{11}$ by one matrix subtraction.
- Step 3: Invert $\mathbf{A}/\mathbf{A}_{11}$ recursively to get the ± 1-values of the remaining $\frac{n}{2}$ diagonal variables and $(\mathbf{A}/\mathbf{A}_{11})^{-1}$.
- Step 4: Compute $\mathbf{A}^{-1}$ according to the blockwise inversion formula

  $$\begin{bmatrix} \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\left(\mathbf{A}/\mathbf{A}_{11}\right)^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\left(\mathbf{A}/\mathbf{A}_{11}\right)^{-1} \\ -\left(\mathbf{A}/\mathbf{A}_{11}\right)^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \left(\mathbf{A}/\mathbf{A}_{11}\right)^{-1} \end{bmatrix}.$$

  This requires another 3 matrix multiplications

  $$(\mathbf{A}/\mathbf{A}_{11})^{-1}(\mathbf{A}_{21}\mathbf{A}_{11}^{-1}),$$
  $$(\mathbf{A}_{11}^{-1}\mathbf{A}_{12}t)(\mathbf{A}/\mathbf{A}_{11})^{-1},$$
  $$((\mathbf{A}_{11}^{-1}\mathbf{A}_{12})(\mathbf{A}/\mathbf{A}_{11})^{-1})(\mathbf{A}_{21}\mathbf{A}_{11}^{-1})$$

  and also another 3 additions or additive inverses.

The four steps at each recursion level perform 2 inversions of diagonally parametric $\frac{n}{2} \times \frac{n}{2}$ matrices, 6 multiplications and 4 additions or additive inverses of $\frac{n}{2} \times \frac{n}{2}$matrices. Thus,

$$T(n) \le 2T(n/2) + O(n^\omega + n^2) = 2T(n/2) + O(n^\omega).$$

By the master theorem for recurrences (cf. Theorem 4.1 in [5]), $T(n) = O(n^\omega)$. If one is curious about the complexity constant, it is roughly between 6 and 12 times the complexity constant of fast matrix multiplication.

As a running example, consider again the instance with $|\mathbb{F}| = 7$ and the matrix $\mathbf{A}$ in equation (5). Step 1 solves recursively the subproblem instance

$$\mathbf{A}_{11} = \begin{bmatrix} \lambda_0 & 0 & 3 & 0 \\ 0 & \lambda_1 & -2 & 0 \\ 3 & 0 & \lambda_2 & 0 \\ -2 & 0 & 0 & \lambda_3 \end{bmatrix}.$$

and gets the choice $\lambda_i = 1$ for $0 \le i \le 3$ together with

$$\mathbf{A}_{11}^{-1} = \begin{bmatrix} -1 & 0 & 3 & 0 \\ -1 & 1 & -2 & 0 \\ 3 & 0 & -1 & 0 \\ -2 & 0 & -1 & 1 \end{bmatrix}.$$

Then Step 2 generates the second subproblem instance

$$\mathbf{A}/\mathbf{A}_{11} = \begin{bmatrix} \lambda_4 & 0 & 3 & 3 \\ 0 & \lambda_5 & -3 & 2 \\ 0 & 0 & \lambda_6 & -1 \\ 0 & -3 & -1 & \lambda_7 + 3 \end{bmatrix}.$$

Step 3 solves the second instance recursively to get the choice $\lambda_4 = \lambda_5 = \lambda_6 = 1$, $\lambda_7 = -1$, and the corresponding inverse

$$(\mathbf{A}/\mathbf{A}_{11})^{-1} = \begin{bmatrix} 1 & 2 & -1 & 3 \\ 0 & 3 & -2 & 3 \\ 0 & 2 & 3 & 3 \\ 0 & 2 & 2 & 3 \end{bmatrix}.$$

Finally, Step 4 computes $\mathbf{A}^{-1}$ given in equation (6).

Suppose n is not a power of 2. Let $k = 2^{\lceil \log n \rceil} - n$, and $\lambda' = [\lambda'_0, \lambda'_1, \cdots, \lambda'_{k-1}]^\top$ be an auxiliary vector of $k \geq 1$ *dummy* variables (or indeterminates) distinct from λ. We embed $\mathbf{A}$ as a block into the block diagonal matrix

$$\mathbf{A}' = \begin{bmatrix} diag(\lambda') & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix}.$$

For any $\lambda' \in \{\pm 1\}^k$ and $\lambda \in \{\pm 1\}^n$ such that $\mathbf{A}'$ is invertible, both $diag(\lambda')$ and $\mathbf{A}$ are invertible and the inverse of $\mathbf{A}'$ is

$$\begin{bmatrix} (diag(\lambda'))^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{-1} \end{bmatrix}.$$

Apply the D&C inversion algorithm to $\mathbf{A}'$, and then extract from the output a solution for inverting $\mathbf{A}$. This at most doubles the dimension and thus increases the complexity constant inside $O(n^\omega)$ by at most a factor of 2^ω. Thus $T(n)$ is $O(n^\omega)$ for all $n \geq 1$.

The theorem below summarizes the performance of the D&C inversion algorithm.

Theorem 3.2: The D&C inversion algorithm inverts a diagonally parametric $n \times n$ matrix $\mathbf{A}$ in $O(n^\omega)$ time.

Thus, the deterministic inversion of diagonally parametric matrices enables sub-cubic-time synthesis of the encoder transformation used in the distributed secret-sharing scheme.

IV. Encoding and decoding scheme

In this section, we consider an arbitrary integral member r of the capacity region, and devise the encoding and decoding scheme for secret messages with rate tuple r. The $r(U)$ secret symbols are uniformly distributed in $\mathbb{F}$ with $|\mathbb{F}| > \Delta$ and mutually independent. In addition, we choose $|V| - r(U)$ *noise* symbols from $\mathbb{F}$ independently and uniformly at random, which are also independent with the $r(U)$ secret symbols. These $|V|$ secret and noise symbols will be both necessary and sufficient to encode $|V|$ shares which are uniformly distributed in $\mathbb{F}$ and mutually independent. Conceptually, the encoding and decoding scheme is constructed in three stages:

- Stage 1: Enumerate nodes and edges according to a perfect r-star matching.
- Stage 2: Construct local decodings by forming local interpolation polynomials and interpolation pairs for each user.
- Stage 3: Recover shares by inverting a diagonally parametric matrix.

These three stages are presented in the first three subsections respectively. The weak privacy of the encoding and decoding scheme is asserted in the last subsection.

Throughout this section, we fix a primitive element γ in $\mathbb{F}$, i.e. each non-zero element of $\mathbb{F}$ is a power of γ. For example, for $|\mathbb{F}| = 7$, $\gamma = 3$ is a primitive element in $\mathbb{F}$ with

$$\begin{bmatrix} \gamma^0 & \gamma^1 & \gamma^2 & \gamma^3 & \gamma^4 & \gamma^5 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 2 & -1 & -3 & -2 \end{bmatrix}.$$

For any vector y on $\mathbb{F}$ indexed by the storage nodes in V and any $S \subseteq V$, y_S denotes the subvector of y indexed by storage nodes in S.

A. Enumerating nodes and edges

In Stage 1, we enumerate the nodes and edges properly to ensure logical consistency between local decodings and global encoding. This enumeration is also essential to forming a diagonally parametric matrix in Stage 3.

The enumeration starts with a perfect r-star matching, whose existence is guaranteed by Theorem 2.2. We first compute a perfect r-star matching M in $O(\min\{|U|, r(U)^{1/2}\}\,|E|)$ time by applying the flow-based algorithm in Section II. Then we extend M to a maximum star matching M^* with $|M^*| = |V|$ edges by picking an arbitrary mate for each storage node unmatched by M. This extension takes $O(|E|)$ time. The $|V|$ edges in M^* are called *star* edges, and the rest $|E| - |V|$ edges are called *non-star* edges. For each user u, let r_u^* denote the number of star edges incident to u, and $N^*(u)$ be the set of storage endpoints of these edges. Clearly, $r_u \leq r_u^* \leq |N(u)|$. In addition, $r^*(U) = |M^*| = |V|$. Now, we enumerate all star edges in the increasing order of user endpoint, breaking ties arbitrarily. Accordingly, we renumber the storage nodes such that for each $j \in [0 : |V| - 1]$, the storage endpoint of the j-th star edge is renumbered by j. This renumbering takes $O(|V|)$ time. Figure 6 illustrates an example of renumbering starting with the perfect r-star matching shown in Figure 2.

Next, at each user u we enumerate all the $|N(u)|$ edges incident to u as follows:

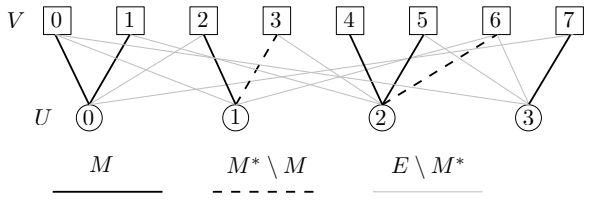

Fig. 6. Renumbering storage nodes according to a user-major order of star edges.

- first all $|N(u)|-r_u^*$ non-star edges in the *increasing* order of the storage endpoint, and
- then all r_u^* star edges in the *increasing* order of the storage endpoint.

At each user u, let $\pi_u(i)$ be the index of the storage endpoint of the i-th edge for $0 \le i \le |N(u)| - 1$. Figure 7 provides the local enumeration of user 1's neighborhood in the running example shown in Figure 6. The local enumerations of all star edges can be derived directly from the global enumeration of all star edges. The local enumerations of all non-star edges can be collectively constructed in $O(|E|)$ time as follows. We scan the storage nodes in the *increasing* order. When a node $v \in V$ is scanned, for each non-star edge (u, v) we place v at the *next* vacant position in the list π_u. The time taken by scanning of v is linear in the number of neighbors of v. Hence, the total time is $O(|E|)$.

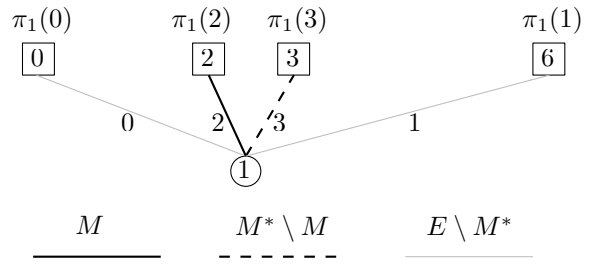

Fig. 7. Local enumeration of user 1's neighborhood.

The time complexity of Stage 1 is

$$O(\min\{|U|, r(U)^{1/2}\}\,|E|),$$

dominated by the computation of the perfect r-star matching M. Once the renumbering of all storage nodes and the local enumerations of neighboring storage nodes by all users are fixed, we move onto the next stage of local decodings.

B. Local decoding

In Stage 2, local decodings are based on multipoint polynomial interpolation. The local decoding at each user u requires

- the forming of local interpolation polynomial $q_u(x)$ of degree less than $|N(u)|$, and
- the specification of $|N(u)|$ interpolation pairs of point and value with *distinct* points.

The system of local interpolation equations at u is then built up from them. We elaborate on the local decodings below.

Each user u will receive $|N(u)| - r_u^*$ dummy symbols and $r_u^* - r_u$ noise symbols exclusively, in addition to its own r_u secret symbols. These $|N(u)|$ symbols form the local interpolation polynomial $q_u(x)$ as coefficients of terms in the ascending order of exponent. Specifically, let X_u denote the column vector

$$\begin{bmatrix} X_{u,0} & X_{u,1} & \cdots & X_{u,r_u^*-1} \end{bmatrix}^\top$$

formed by its $r_u^* - r_u$ noise symbols and then its own r_u secret symbols *successively*, and let Z_u denote the column vector

$$\begin{bmatrix} Z_{u,0} & Z_{u,1} & \cdots & Z_{u,|N(u)|-r_u^*-1} \end{bmatrix}^\top$$

formed by its $|N(u)| - r_u^*$ dummy symbols. Then

$$q_u(x) = p_{Z_u}(x) + x^{|N(u)|-r_u^*} p_{X_u}(x),$$

where $p_{Z_u}(x)$ interpolates dummy symbols and $p_{X_u}(x)$ interpolates noise/secret symbols (cf. the notation in equation (4)). For the running example in Figure 6, the four local polynomials are

$$\begin{aligned}
q_0(x) &= (Z_{0,0} + Z_{0,1}x) + x^2(X_{0,0} + X_{0,1}x),\\
q_1(x) &= (Z_{1,0} + Z_{1,1}x) + x^2(X_{1,0} + X_{1,1}x),\\
q_2(x) &= (Z_{2,0} + Z_{2,1}x) + x^2(X_{2,0} + X_{2,1}x + X_{2,2}x^2),\\
q_3(x) &= (Z_{3,0} + Z_{3,1}x + Z_{3,2}x^2) + x^3(X_{3,0}).
\end{aligned}$$

Now we specify all interpolation pairs explicitly. Technically at each user u, every edge (u, v) supplies an interpolation pair of point and value, where

- the interpolation point is a power of the primitive element γ, and
- the interpolation value is a *scaled share*, the product of the *share* at v and the *scaling factor* of the edge (u, v).

The $|E|$ scaling factors are specified based on the star matching M^*:

- Each non-star edge has a *fixed unit* scaling factor.
- Each star edge (u, v) has a *parametric* scaling factor $\lambda_v \in \{\pm 1\} \subset \mathbb{F}$. It is well-defined because each storage node $v \in V$ is incident to a unique star edge. Thus, λ_v is also referred to as the scaling factor of v. The choice of the binary signs later by global encoding will ensure invertibility of local interpolation systems.

Let Y (respectively, λ) be the column vector of $|V|$ share symbols (respectively, scaling factors) indexed by the storage nodes in V, both of which will be determined in Stage 3. Then the $|N(u)|$ interpolation pairs for user u are

$$\begin{aligned}
&\left(\gamma^i, Y_{\pi_u(i)}\right) \text{ for } 0 \le i \le |N(u)| - r_u^* - 1;\\
&\left(\gamma^i, \lambda_{\pi_u(i)} Y_{\pi_u(i)}\right) \text{ for } |N(u)| - r_u^* \le i \le |N(u)| - 1.
\end{aligned}$$

For the running example shown in Figure 6, the interpolation pairs for user 0, 1, 2, and 3 are respectively listed below:

$$(\gamma^0, Y_2), (\gamma^1, Y_7), (\gamma^2, \lambda_0 Y_0), (\gamma^3, \lambda_1 Y_1);$$
$$(\gamma^0, Y_0), (\gamma^1, Y_6), (\gamma^2, \lambda_2 Y_2), (\gamma^3, \lambda_3 Y_3);$$
$$(\gamma^0, Y_1), (\gamma^1, Y_3), (\gamma^2, \lambda_4 Y_4), (\gamma^3, \lambda_5 Y_5), (\gamma^4, \lambda_6 Y_6);$$
$$(\gamma^0, Y_0), (\gamma^1, Y_5), (\gamma^2, Y_6), (\gamma^3, \lambda_7 Y_7).$$

For each user u, its local interpolation polynomial and $|N(u)|$ interpolation pairs yields its system of $|N(u)|$ interpolation equations:

$$q_u\left(\gamma^i\right) = Y_{\pi_u(i)} \text{ for } 0 \le i \le |N(u)| - r_u^* - 1; \quad (7)$$
$$q_u\left(\gamma^i\right) = \lambda_{\pi_u(i)} Y_{\pi_u(i)} \text{ for } |N(u)| - r_u^* \le i \le |N(u)| - 1. \quad (8)$$

Given the subvector $Y_{N(u)}$ of shares and the subvector $\lambda_{N(u)}$ of scaling factors, all symbols in X_u and Z_u can be decoded by multipoint polynomial interpolation in $O(|N(u)| \log^2 |N(u)|)$ time.

Stage 2 only defines the local decoding scheme, and does not incur any computation cost. The computation of Y and λ occurs in Stage 3, to which we proceed next.

C. Global encoding

Global encoding in Stage 3 selects the scaling vector $\lambda \in \{\pm 1\}^{|V|}$ and encodes the share vector Y from the secret and noise symbols. Let X be the column vector formed by stacking the vectors X_u in the increasing order of u. The three vectors X, Y, and λ all have $|V|$ entries. Global encoding is performed in two steps:

- **Step 1**: Form the $|V|$ dummy-free equations in matrix form $\mathbf{A}Y = \mathbf{B}X$ from the system of $|E|$ interpolation equations (7) and (8). The matrix $\mathbf{A}$ is a diagonally parametric square matrix with λ as the diagonal vector, and the matrix $\mathbf{B}$ is plain square matrix without parameters.
- **Step 2**: Invert the parametric matrix $\mathbf{A}$ to get an invertible choice of scaling vector $\lambda \in \{\pm 1\}^{|V|}$ and $\mathbf{A}^{-1}$, and output the encoding scheme $Y = \mathbf{A}^{-1}\mathbf{B}X$.

It's important to observe that λ and $\mathbf{A}^{-1}\mathbf{B}$ depend only on the rate vector r but not on the secret/noise vector X. We remark that we shall not compute the matrix $\mathbf{A}^{-1}\mathbf{B}$ to avoid the computationally expensive matrix multiplication. Instead, from any given X, the encoding computes Y by two matrix-vector multiplications in $O(|V|^2)$ time: first compute $\mathbf{B}X$, and then compute $\mathbf{A}^{-1}(\mathbf{B}X)$. Thus, the encoding time complexity is $O(|V|^2)$. In the sequel, we describe the implementation of the two steps.

Step 1 will get rid of dummy variables through Gaussian eliminations. Its implementation relies on fast Block LU Decompositions of Fourier matrices. For any positive integer $n < |\mathbb{F}|$, the *Fourier matrix* of order n is the matrix $\left[\gamma^{ij}\right]_{i,j=0}^{n-1}$, or explicitly:

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \gamma^1 & \gamma^2 & \cdots & \gamma^{n-1} \\ 1 & \gamma^2 & \gamma^4 & \cdots & \gamma^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma^{n-1} & \gamma^{2(n-1)} & \cdots & \gamma^{(n-1)^2} \end{bmatrix}.$$

The Fourier matrix $\mathbf{F}$ is a symmetric Vandermonde matrix generated by $\left[\gamma^i\right]_{i=0}^{n-1}$. The Block LU Decompositions of $\mathbf{F}$ can be efficiently computed by multipoint polynomial interpolation and evaluation as asserted in the theorem below.

Theorem 4.1: For a Fourier matrix $\mathbf{F}$ of order $n \in [2 : |\mathbb{F}|]$ with block partition

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} \\ \mathbf{F}_{21} & \mathbf{F}_{22} \end{bmatrix}$$

where $\mathbf{F}_{11}$ has dimension $k \times k$, both $\mathbf{F}_{21}\mathbf{F}_{11}^{-1}$ and $\mathbf{F}/\mathbf{F}_{11}$ can be computed in $O((n-k)\, n \log^2 n)$ time.

Proof. If $k = 1$, then $\mathbf{F}_{21}\mathbf{F}_{11}^{-1}$ is the all-1 vector, and $\mathbf{F}_{21}\mathbf{F}_{11}^{-1}\mathbf{F}_{12}$ is the all-1 matrix; hence both $\mathbf{F}_{21}\mathbf{F}_{11}^{-1}$ and $\mathbf{F}/\mathbf{F}_{11}$ can be trivially computed in $O((n-k)\, n)$ time. Henceforth, we assume that $k > 1$.

We compute both $\mathbf{F}_{21}\mathbf{F}_{11}^{-1}$ and $\mathbf{F}/\mathbf{F}_{11}$ row by row with $O(n \log^2 n)$ time complexity per row, implying the overall $O((n-k)\, n \log^2 n)$ time complexity. Consider any row i with $0 \le i \le n-k-1$. Denote the row-i vector of $\mathbf{F}_{21}\mathbf{F}_{11}^{-1}$ by a. Then $a\mathbf{F}_{12}$ is the the row-i vector of $\mathbf{F}_{21}\mathbf{F}_{11}^{-1}\mathbf{F}_{12}$. The polynomial $p_{a^\tau}(x)$ has degree less than k. The computation proceeds in three steps.

Step 1: Compute a by multipoint polynomial *interpolation*. Since $a\mathbf{F}_{11}$ is the row-i vector of $\mathbf{F}_{21}$, the values of the polynomial $p_{a^\tau}(x)$ at γ^j for $j = 0, 1, \cdots, k-1$ are exactly the entries in the row-i vector of $\mathbf{F}_{21}$. Hence, a can be computed via multipoint polynomial interpolation in $O(k \log^2 k)$ time.

Step 2: Compute $a\mathbf{F}_{12}$ by multipoint polynomial *evaluation*. Evaluating the polynomial $p_{a^\tau}(x)$ at γ^j for $j = k, k+1, \cdots, n-1$ yields the entries in $a\mathbf{F}_{12}$. Hence, $a\mathbf{F}_{12}$ can be computed via multipoint polynomial evaluation in

$$O((k + (n-k)) \log^2(k + (n-k))) = O(n \log^2 n)$$

time.

Step 3: Compute the row-i vector of $\mathbf{F}/\mathbf{F}_{11}$ by subtracting $a\mathbf{F}_{12}$ from the row-i vector of $\mathbf{F}_{22}$. This simple step takes $O(n-k)$ time.

The total time complexity of the three steps is thus $O(n \log^2 n)$. ■

Now we implement **Step 1** by repeated applications of Theorem 4.1. **Step 1** eliminates the $|E| - |V|$ dummy symbols in Z from the system of $|E|$ interpolation equations (7) and (8) locally at each user u. We skip those users u with $r_u^* = 0$ as they supply no secret or noise symbol. For each user u with $r_u^* > 0$, let $\mathbf{F}$ be the Fourier matrix of order $|N(u)|$ and

we eliminate the $|N(u)| - r_u^*$ dummy symbols in Z_u from the $|N(u)|$ interpolation equations (7) and (8) in two cases.

Case 1: $|N(u)| = r_u^*$. Then $N(u) = N^*(u)$ and there is no dummy variable in Z_u. The $r_u^* = |N^*(u)|$ interpolation equations (7) and (8) has the matrix form

$$diag(\lambda_{N^*(u)})\, Y_{N^*(u)} = \mathbf{F} X_u. \tag{9}$$

Case 2: $|N(u)| > r_u^* > 0$. We conformally partition the matrix $\mathbf{F}$ into blocks

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} \\ \mathbf{F}_{21} & \mathbf{F}_{22} \end{bmatrix}$$

such that $\mathbf{F}_{22}$ has dimension $r_u^* \times r_u^*$. Then, the $|N(u)|$ interpolation equations (7) and (8) has the matrix form

$$\begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} \\ \mathbf{F}_{21} & \mathbf{F}_{22} \end{bmatrix} \begin{bmatrix} Z_u \\ X_u \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{|N(u)|-r_u^*} & \mathbf{0} \\ \mathbf{0} & diag(\lambda_{N^*(u)}) \end{bmatrix} \begin{bmatrix} Y_{N(u)\setminus N^*(u)} \\ Y_{N^*(u)} \end{bmatrix}.$$

Left-multiplying both sides by the (Gaussian elimination) matrix

$$\begin{bmatrix} \mathbf{I}_{|N(u)|-r_u^*} & \mathbf{0} \\ -\mathbf{F}_{21}\mathbf{F}_{11}^{-1} & \mathbf{I}_{r_u^*} \end{bmatrix}$$

yields

$$\begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} \\ \mathbf{0} & \mathbf{F}/\mathbf{F}_{11} \end{bmatrix} \begin{bmatrix} Z_u \\ X_u \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{|N(u)|-r_u^*} & \mathbf{0} \\ -\mathbf{F}_{21}\mathbf{F}_{11}^{-1} & diag\left(\lambda_{N^*(u)}\right) \end{bmatrix} \begin{bmatrix} Y_{N(u)\setminus N^*(u)} \\ Y_{N^*(u)} \end{bmatrix}.$$

The identity on the second row of blocks gives r_u^* dummy-free equations in the matrix form

$$\left[diag(\lambda_{N^*(u)}) \quad -\mathbf{F}_{21}\mathbf{F}_{11}^{-1}\right] \begin{bmatrix} Y_{N^*(u)} \\ Y_{N(u)\setminus N^*(u)} \end{bmatrix} = \mathbf{F}/\mathbf{F}_{11} \cdot X_u. \tag{10}$$

By Theorem 4.1, both $-\mathbf{F}_{21}\mathbf{F}_{11}^{-1}$ and $\mathbf{F}/\mathbf{F}_{11}$ can be computed in $O(r_u^* \Delta \log^2 \Delta)$ time.

Thus, each user u generates r_u^* dummy-free equations in (9) or (10), yielding

$$\textstyle\sum_{u\in U} r_u^* = r^*(U) = |V|$$

dummy-free equations in total. We assemble those $|V|$ dummy-free equations in the matrix form $\mathbf{A}Y = \mathbf{B}X$. As each λ_v appears exactly once as a coefficient of Y_v in the dummy-free equations, the matrix $\mathbf{A}$ is a diagonally parametric matrix. The matrix $\mathbf{B}$ is a block-diagonal matrix, where each diagonal block is of the form $\mathbf{F}/\mathbf{F}_{11}$. Thus $\mathbf{B}$ is invertible. The total computation time taken by the eliminations for all users u in U is

$$O\left(\textstyle\sum_{u\in U} r_u^* \Delta \log^2 \Delta\right) = O(|V| \Delta \log^2 \Delta).$$

Assembling those $|V|$ dummy-free equations takes additional $O(|V|^2)$ time. So, the total running time of **Step 1** is $O(|V| \Delta \log^2 \Delta + |V|^2)$.

We continue to **Step 2** on inverting the diagonally parametric matrix **A.** This step is implemented simply by applying the D&C inversion algorithm in Subsection III-B to the matrix $\mathbf{A}$. Thus, in $O(|V|^\omega)$ time **Step 2** chooses $\lambda \in \{\pm 1\}^{|V|}$ such that the matrix $\mathbf{A}$ is invertible and computes $\mathbf{A}^{-1}$ at such λ.

As $\Delta \le |V|$ and $\omega > 2$, the $O(|V| \Delta \log^2 \Delta + |V|^2)$ time complexity of **Step 1** is dominated asymptotically by the $O(|V|^\omega)$ time complexity of **Step 2**. Thus, the time complexity of Stage 3 is $O(|V|^\omega)$. In appendix, we illustrate the execution of Stage 3 on the running example in Figure 6 with $|\mathbb{F}| = 7$ and $\gamma = 3$.

This completes the 3-stage construction of the encoding and decoding scheme. The overall time complexity of the three stages is $O(\min\{|U|, r(U)^{1/2}\}\,|E| + |V|^\omega)$. The weak privacy of the resulting encoder/decoder is asserted in the next subsection.

D. Weak privacy

After placing shares at storage nodes, the dealer only needs to inform the users of the following $O(|V|)$ values sufficient for decoding by the users:

- $|\mathbb{F}|$ and the primitive element γ;
- the star matching M^* (and M if each user u does not know its rate r_u);
- the renumbering of the storage nodes;
- the indices v of the storage nodes with scaling factor $\lambda_v = -1$.

Despite that each user can access all shares at neighboring storage nodes, we assert the weak privacy of our encoding and decoding scheme in this subsection.

For each sequence Q of random variables on $\mathbb{F}$, $H(Q)$ denotes its Shannon entropy. As the $|V|$ symbols in X are chosen independently and uniformly at random from $\mathbb{F}$, $H(X) = |V| \log |\mathbb{F}|$. Since X and Y can be derived from each other, both $H(X \mid Y)$ and $H(Y \mid X)$ are 0; hence

$$H(Y) = H(X,Y) = H(X) = |V| \log |\mathbb{F}|\,.$$

Thus, the $|V|$ shares in Y also have full entropy and are mutually independent.

For each user u, let W_u be the vector of its r_u secret symbols. Then $H(W_u) = r_u \log |\mathbb{F}|$. The following theorem on weak privacy follows from an argument similar to that in Section IV.E of [13].

Theorem 4.2: For any two distinct users u and u', $H(W_u \mid Y_{N(u')}) = H(W_u)$.

Proof. By the private-degree constraint on r (cf. equation (2)),

$$r_u \le b_u \le |N(u) \setminus N(u')|\,;$$

hence $N(u) \setminus N(u')$ has a subset T with $|T| = r_u$. From W_u and $Y_{V\setminus T}$, we can derive Y_T in two steps:

- Step 1: From the $|N(u) \setminus T| = |N(u)| - r_u$ interpolation equations (7) and (8) at u with $\pi_u(i) \in N(u) \setminus T$, we calculate the $|N(u)| - r_u$ dummy and noise symbols distributed to u. This is possible because the coefficient matrix of these symbols is the Vandermonde submatrix generated by $\left[\gamma^i\right]_{\pi_u(i)\in N(u)\setminus T}$ and is thus invertible.
- Step 2: From the rest $|T|$ interpolation equations (7) and (8) at u with $\pi_u(i) \in T$, we calculate the $|T|$ share symbols in Y_T. This is possible because the coefficient of each share symbol is *non-zero*.

Thus,

$$H(Y_T \mid W_u, Y_{V\setminus T}) = 0.$$

As $N(u') \subseteq V \setminus T$, the above equality together with the mutual independence of the $|V|$ shares in Y implies that

$$\begin{aligned}
&H(W_u) \geq H(W_u \mid Y_{N(u')}) \geq H(W_u \mid Y_{V\setminus T}) \\
&= H(W_u \mid Y_{V\setminus T}) + H(Y_T \mid W_u, Y_{V\setminus T}) = H(W_u, Y_T \mid Y_{V\setminus T}) \\
&\geq H(Y_T \mid Y_{V\setminus T}) = H(Y_T) = |T| \log|\mathbb{F}| = r_u \log|\mathbb{F}| \\
&= H(W_u).
\end{aligned}$$

Thus, all inequalities hold with equality throughout; hence the theorem follows. ■

Hence, the deterministic encoding and decoding process achieves all integral capacity points efficiently, closing the gap between theoretical feasibility and explicit construction.

V. Conclusion

This work establishes that deterministic algebraic constructions can achieve the full capacity region of weakly private distributed multi-user secret sharing with polynomial-time complexity. Two key algorithmic primitives have been introduced in Section II and Section III respectively:

- a combinatorial method for finding a maximum r-star matching, and
- an algebraic technique for inverting diagonally parametric matrices.

Together, these yield a fully deterministic encoder/decoder whose computational complexity is orders of magnitude lower than the prior randomized scheme, while requiring $|\mathbb{F}| > \Delta$ (the maximum user degree). These deterministic constructions may also inform explicit coded-computing, distributed-storage, and cryptographic protocol design where reproducibility and finite-field determinism are essential.

This work, however, is subject to open practical constraints: finite-field size requirement $|\mathbb{F}| > \Delta$, potential numerical instability in large-field arithmetic, and lack of experimental benchmarking. Nonetheless, this work has paved the way for future studies of (fully deterministic) algorithmic problems over the capacity region of weakly private distributed multi-user secret sharing. We give two open problems below.

The first open problem is convex decomposition of a non-integral member of the capacity region into integral members of the capacity region. The existence of such convex decomposition is ensured by Carathéodory's theorem. However, little is known about the algorithmic version of Carathéodory's theorem for the capacity region. Efficient convex decomposition in high-dimensional capacity regions is computationally challenging due to exponential extreme-point enumeration. The exact link between integral rate tuples and feasible flow configurations established in this paper enables a flow-based approach for which a plenty of algorithmic tools [1] are available. We are currently developing an efficient convex decomposition algorithm following this approach.

The second open problem is rate allocation subject to user-specific requirement with application-specific objectives. Representative objectives include max-min fairness, proportional fairness, and total throughput maximization. Each optimization criterion leads to a different convex-optimization formulation over the capacity region. Efficient algorithms for those optimization problems may require creative applications of techniques from submodular optimizations [9] and network flows [1].

Acknowledgments The authors thank the anonymous referees for their generous, insightful, comprehensive, and constructive comments. Following their suggestions and advices, the paper has been significantly improved in both rigor and exposition from the first submission.

Appendix

In this appendix, we illustrate the execution of Stage 3 on the running example in Figure 6 with $|\mathbb{F}| = 7$ and $\gamma = 3$.

The execution of **Step 1** is outlined as follows. At user 0, the conformal partition of the Fourier matrix of order 4 is

$$\left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & 3 & 2 & -1 \\ \hline 1 & 2 & -3 & 1 \\ 1 & -1 & 1 & -1 \end{array}\right].$$

As

$$\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} -2 & 3 \\ 3 & -3 \end{bmatrix},$$

the dummy-free equation of user 0 is

$$\begin{bmatrix} \lambda_0 & 0 & 3 & 3 \\ 0 & \lambda_1 & -2 & 1 \end{bmatrix} \begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_7 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} X_{0,0} \\ X_{0,1} \end{bmatrix}.$$

At user 1, the *same* elimination process at user 0 yields the dummy-free equation of user 1:

$$\begin{bmatrix} \lambda_2 & 0 & 3 & 3 \\ 0 & \lambda_3 & -2 & 1 \end{bmatrix} \begin{bmatrix} Y_2 \\ Y_3 \\ Y_0 \\ Y_6 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} X_{1,0} \\ X_{1,1} \end{bmatrix}$$

At user 2, the conformal partition of the Fourier matrix of order 5 is

$$\left[\begin{array}{cc|ccc} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 2 & -1 & -3 \\ \hline 1 & 2 & -3 & 1 & 2 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & -3 & 2 & 1 & -3 \end{array}\right].$$

Similarly, the dummy-free equation of user 2 is

$$\begin{bmatrix} \lambda_4 & 0 & 0 & 3 & 3 \\ 0 & \lambda_5 & 0 & -2 & 1 \\ 0 & 0 & \lambda_6 & -3 & 2 \end{bmatrix} \begin{bmatrix} Y_4 \\ Y_5 \\ Y_6 \\ Y_1 \\ Y_3 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 3 \\ 1 & 3 & 3 \\ 3 & 3 & 2 \end{bmatrix} \begin{bmatrix} X_{2,0} \\ X_{2,1} \\ X_{2,2} \end{bmatrix}$$

At user 3, the conformal partition of the Fourier matrix of order 4 is

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 1 & 3 & 2 & -1 \\ 1 & 2 & -3 & 1 \\ \hline 1 & -1 & 1 & -1 \end{array}\right].$$

As

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 2 \\ 1 & 2 & -3 \end{bmatrix}^{-1} = \begin{bmatrix} 3 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & -1 \end{bmatrix},$$

the dummy-free equation of user 3 is

$$\begin{bmatrix} \lambda_7 & 1 & -3 & 1 \end{bmatrix} \begin{bmatrix} Y_7 \\ Y_0 \\ Y_5 \\ Y_6 \end{bmatrix} = -3X_{3,0}.$$

Combining the dummy-free equations of all four users, we get the matrix form $\mathbf{A}Y = \mathbf{B}X$, where $\mathbf{A}$ is given by equation (5) and

$$\mathbf{B} = \left[\begin{array}{cc|cc|cc|c} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & -1 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 & 3 & 3 & 2 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 \end{array}\right].$$

The execution of **Step 2** is already illustrated in the running example in Subsection III-B. It chooses $\lambda_i = 1$ for $0 \le i \le 6$ and $\lambda_7 = -1$, and outputs $\mathbf{A}^{-1}$ given by equation (6). The matrix $\mathbf{A}^{-1}\mathbf{B}$, while not required, is the coefficient matrix in equation (1).

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.

[2] J. Alman, R. Duan, V. V. Williams, Y. Xu, Z. Xu, and R. Zhou, More asymmetry yields faster matrix multiplication. *Proc. SODA 2025*: 2005–2039, 2025.

[3] N. Alon, Combinatorial Nullstellensatzm, *Combinatorics, Probability and Computing* 8(1–2): 7–29, 1999.

[4] E. Ben-Sasson, D. Carmon, S. Kopparty, and D. Levit, Elliptic Curve Fast Fourier Transform (ECFFT) Part I: Low-degree extension in time $O(n \log n)$ over all finite fields, *Proc. SODA 2023*: 700–737, 2023.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (4th Edition), MIT Press, 2022.

[6] E. A. Dinitz, Algorithm for solution of a problem of maximum flow in a network with power estimation, *Soviet Mathematics Doklady* 11: 1277–1280, 1970.

[7] A. Frank and E. Tardos, Generalized polymatroids and submodular flows, *Mathematical Programming* 42: 489–563, 1988.

[8] S. Even and R. E. Tarjan, Network flow and testing graph connectivity, *SIAM Journal on Computing* 4(4): 507–518, 1975.

[9] S. Fujishige, *Submodular Functions and Optimization* (2nd Edition), Annals of Discrete Mathematics vol. 58, Elsevier, Amsterdam, 2005.

[10] T. Helgason, Aspects of the theory of hypermatroids, *Hypergraph Seminar*, Springer Lecture Notes in Mathematics 411: 191–213, 1974.

[11] J. Herzog and T. Hibi, Discrete polymatroids, *Journal of Algebraic Combinatorics* 16: 239–268, 2002.

[12] J. E. Hopcroft and R. M. Karp, An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, *SIAM Journal on Computing* 2: 225–231, 1973.

[13] A. Khalesi, M. Mirmohseni, and M. A. Maddah-Ali, The capacity region of distributed multi-user secret sharing, *IEEE Journal on Selected Areas in Information Theory* 2(3): 1057–1071, 2021. DOI: 10.1109/JSAIT.2021.3102967.

[14] A. Schrijver, *Combinatorial Optimization*, Springer, Berlin, 2003.

[15] J. T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, *Journal of the ACM* 27 (4): 701–717, 1980.

[16] A. Shamir, How to share a secret, *Communications of the ACM* 22(11): 612–613, 1979.

[17] C. E. Shannon, Communication theory of secrecy systems, *Bell System Technical Journal* 28(4): 656–715, 1949.

[18] M. Soleymani and H. Mahdavifar, Distributed multi-user secret sharing, *IEEE Transactions on Information Theory* 67(1): 164–178, 2021.

[19] V. Strassen, Gaussian elimination is not optimal, *Numerische Mathematik.* 13 (4): 354–356, 1969.

[20] R. Zippel, Probabilistic algorithms for sparse polynomials, *Proc. of EUROSAM'79*, Springer Lecture Notes in Computer Science 72: 216–226, 1979.

PLACE PHOTO HERE

Joy Z. Wan Joy Z. Wan is pursuing the joint B.S./M.S. degree in computer science at the University of Illinois Urbana-Champaign. She is expected to graduate May 2026. This work is part of her M.S. thesis, supervised by Dr. Ning Luo. Her research interests include algorithms, machine learning, and applied cryptography.

PLACE PHOTO HERE

Ning Luo Ning Luo is a tenure-track Assistant Professor in the Department of Electrical and Computer Engineering at the University of Illinois Urbana-Champaign. She received her Ph.D. in Computer Science from Yale University in December 2022 and subsequently spent a year as a postdoctoral researcher at Northwestern University. Ning's research integrates formal methods, automated reasoning, programming languages, and cryptography to achieve security, verifiability, and confidentiality in practical and challenging scenarios.