On Secure Multi-Party Computation in Black-Box Groups*

Yvo Desmedt[1†] and Josef Pieprzyk[2] and Ron Steinfeld[2] and Huaxiong Wang[2,3]

[1] Dept. of Computer Science, University College London, UK
[2] Centre for Advanced Computing – Algorithms and Cryptography (ACAC)
Dept. of Computing, Macquarie University, North Ryde, Australia
[3] Division of Math. Sci., Nanyang Technological University, Singapore
{josef,rons,hwang}@comp.mq.edu.au, hxwang@ntu.edu.sg

Abstract

We study the natural problem of secure n-party computation (in the passive, computationally unbounded attack model) of the n-product function $f_G(x_1, \ldots, x_n) = x_1 \cdot x_2 \cdots x_n$ in an arbitrary finite group $(G, \cdot)$, where the input of party P_i is $x_i \in G$ for $i = 1, \ldots, n$. For flexibility, we are interested in protocols for f_G which require only *black-box* access to the group G (i.e. the only computations performed by players in the protocol are a group operation, a group inverse, or sampling a uniformly random group element).

Our results are as follows. First, on the negative side, we show that if $(G, \cdot)$ is non-abelian and $n \geq 4$, then no $\lceil n/2 \rceil$-private protocol for computing f_G exists. Second, on the positive side, we initiate an approach for construction of black-box protocols for f_G based on k-of-k threshold secret sharing schemes, which are efficiently implementable over any black-box group G. We reduce the problem of constructing such protocols to a combinatorial colouring problem in planar graphs. We then give two constructions for such graph colourings. Our first colouring construction gives a protocol with optimal collusion resistance $t < n/2$, but has exponential communication complexity $O(n\binom{2t+1}{t}^2)$ group elements (this construction easily extends to general adversary structures). Our second probabilistic colouring construction gives a protocol with (close to optimal) collusion resistance $t < n/\mu$ for a graph-related constant $\mu \leq 2.948$, and has efficient communication complexity $O(nt^2)$ group elements. Furthermore, we believe that our results can be improved by further study of the associated combinatorial problems.

Key Words: Multi-Party Computation, Non-Abelian Group, Black-Box, Planar Graph, Graph Colouring.

1 Introduction

Background. Groups form a natural mathematical structure for cryptography. In particular, the most popular public-key encryption schemes today (RSA [20] and Diffie-Hellman/ElGamal [11, 12]) both operate in abelian groups. However, the discovery of efficient quantum algorithms for breaking these cryptosystems [22] gives increased importance to the construction of alternative cryptosystems in non-abelian groups (such as [16, 18]), where quantum algorithms seem to be much less effective.

Motivated by such emerging cryptographic applications of non-abelian groups, we study the natural problem of secure n-party computation (in the passive, computationally unbounded attack model) of the n-product function $f_G(x_1, \ldots, x_n) = x_1 \cdot x_2 \cdots x_n$ in an arbitrary finite group $(G, \cdot)$, where the input of party P_i is $x_i \in G$ for $i = 1, \ldots, n$. For flexibility, we are interested in protocols

*This is the full version of a paper presented at CRYPTO 2007.

†A part of this research was funded by NSF ANI-0087641, EPSRC EP/C538285/1. Yvo Desmedt is BT Chair of Information Security.

for f_G which require only *black-box* access to the group G (i.e. the only computations performed by players in the protocol are a group operation $(x, y) \to x \cdot y$, a group inverse $x \to x^{-1}$, or sampling a random group element $x \in_R G$). It is well known that when $(G, \cdot)$ is abelian, a straightforward 2-round black-box protocol exists for f_G which is t-private (secure against t parties) for any $t < n$ and has communication complexity $O(n^2)$ group elements. However, to our knowledge, when $(G, \cdot)$ is non-abelian, no constructions of black-box protocols for f_G have been designed until now. Consequently, to construct a t-private protocol for f_G in some non-abelian group G one currently has to resort to adopting existing non black-box methods, which may lead to efficiency problems (see 'Related Work').

Our Results. Our results are as follows. First, on the negative side, we show that if $(G, \cdot)$ is non-abelian and $n \geq 4$, then no $\lceil n/2 \rceil$-private protocol for computing f_G exists. Second, on the positive side, we initiate an approach for construction of black-box protocols for f_G based only on k-of-k threshold secret sharing schemes (whereas previous non black-box protocols rely on Shamir's t-of-n threshold secret sharing scheme over a ring). We reduce the problem of constructing such protocols to a combinatorial colouring problem in planar graphs. We then give two constructions for such graph colourings. Our first colouring construction gives a protocol with optimal collusion resistance $t < n/2$, but has exponential communication complexity $O(n\binom{2t+1}{t}^2)$ group elements (this construction also easily generalises to general Q^2 adversary structures $\mathcal{A}$ as defined in [14], giving communication complexity $O(n|\mathcal{A}|^2)$ group elements). Our second probabilistic colouring construction gives a protocol with (close to optimal) collusion resistance $t < n/\mu$ for a graph-related constant $\mu \leq 2.948$, and has efficient communication complexity $O(nt^2)$ group elements. Furthermore, we believe that our results can be improved by further study of the associated combinatorial problems. We note that our protocols easily and naturally generalize to other arbitrary functions defined over the group G.

Related Work. There are two known non black-box methods for constructing a t-private protocol for the n-product function f_G for any $t < n/2$. They are both based on Shamir's t-of-n threshold secret sharing scheme [21] and its generalizations.

The first method [4, 5, 13] requires representing f_G as a boolean circuit, and uses Shamir's secret sharing scheme over the field $GF(p)$ for a prime $p > 2t + 1$. This protocol has total communication complexity $O(t^2 \log t \cdot N_{AND}(f_G))$ bits, where $N_{AND}(f_G)$ denotes the number of AND gates in the boolean AND/NOT circuit for computing f_G. Thus this protocol is efficient only for very small groups G, for which $N_{AND}(f_G)$ is manageable.

The second method [8] (see also [2] for earlier work) requires representing f_G as an arithmetic circuit over a finite *ring* R, and accordingly, uses a generalization of Shamir's secret sharing scheme to any finite ring. This protocol has total communication complexity $O(t^2 \log t \cdot N_M(f_G) \cdot \ell(R))$ bits, where $N_M(f_G)$ is the number of multiplication operations in the circuit for f_G over R and $\ell(R) \geq \log |R|$ denotes the number of bits needed for representing elements of R. If we 'embed' group G in the ring $R = R(G)$, so that R inherits the multiplication operation of G, then $N_M(f_G) = n - 1$, and hence the protocol from [8] has total communication complexity $O(nt^2 \log t \cdot \ell(R(G)))$ bits, compared to $O(nt^2 \cdot \ell(G))$ bits for our (second) protocol (assuming $t < n/2.948$), where $\ell(G) \geq \log |G|$ is the representation length of elements of G. Hence, for $t < n/2.948$, the communication complexity of our protocol for f_G is smaller than the one from [8] by a factor $\Theta(\frac{\ell(R(G))}{\ell(G)} \cdot \log t)$ (for $n/2.948 < t < n/2$, the protocol of [8] is still asymptotically the most efficient known proven protocol). Note that, for any finite group G, we can always take $R(G)$ to be the *group algebra* (or group ring) of G over $GF(2)$, which can be viewed as a $|G|$-dimensional vector space over $GF(2)$ consisting of all linear combinations of the elements of G (the basis vectors) with coefficients from $GF(2)$ (the product operation of $R(G)$ is defined by the operation of G extended by linearity and associativity, and the addition operation of $R(G)$ is defined componentwise). However, for this generic choice of $R(G)$ we have $\ell(R(G)) = |G|$, so, assuming $\ell(G) = \log |G|$, our protocol reduces communication complexity by a factor $\Theta(\frac{|G|}{\log |G|} \cdot \log t)$, which is exponentially large in the representation length $\log |G|$. In the

worst case, we may have $\ell(R(G)) = \Theta(\ell(G))$ and our protocol may only give a saving factor $O(\log t)$ over the protocol from [8], e.g. this is the case for $G = GL(k, 2)$ (the group of invertible $k \times k$ matrices over $GF(2)$). We remark that this $O(\log t)$ saving factor arises essentially from the fact that Shamir's secret sharing for $2t + 1$ shares requires a ring of size greater than $2t + 1$, and hence, for a secret from $GF(2)$, the share length is greater than the secret length by a factor $\Theta(\log t)$ (whereas our approach does not use Shamir's sharing and hence does not suffer from this length expansion). On the other hand, for sharing a secret from $GF(q)$ for 'large' q ($q > 2t + 1$), Shamir's scheme is ideal, so for specific groups such as $G = GL(k, q)$ with $q > 2t + 1$, the communication cost of the protocols from [2, 8] reduces to $O(nt^2 \cdot \ell(R(G)))$.

Application to General Multi-Party Computation. After the presentation of this paper at CRYPTO 2007, we became aware of a result due to Barrington [3], which implies that our black-box group multiplication protocols, when applied to the symmetric group S_5, can be used as a building block to perform multi-party computation of *arbitrary* functions, secure against passive adversaries (We refer the reader to the end of Sec. 4.5 for more details). Namely, in the paper [3], it shown that any Boolean AND/NOT circuit C (containing N_{AND} 2-input AND gates) can be converted into an algebraic circuit C' over the group S_5 (containing $O(N_{AND})$ 2-input S_5 multiplication gates), such that C' computes the same Boolean function as C, when using a certain encoding of Boolean values using S_5 elements. Our t-private protocols for computing the n-Product function f_G (implemented with $G = S_5$) are easily generalized to give t-private protocols for computing the algebraic circuit C' (applying our 'Shared 2-Product Subprotocol' at each S_5 multiplication gate in circuit C'). Since the order $|S_5| = 120$ of group S_5 is a small constant independent of the number of players n, each share in our protocols can be encoded using a constant length string (of at most 7 bits). Similarly to the discussion above for the case of f_G, the resulting communication complexity of our 'probabilistic colouring' protocol for circuit C is $O(N_{AND} \cdot t^2)$ bits, a saving of an $O(\log t)$ factor over the Shamir-based protocol, but at the expense of a relaxed privacy threshold $t < n/2.948$. Interestingly, similar tradeoffs of privacy versus communication complexity for multiparty computation have recently been independently shown via a different approach, namely generalizations of Shamir secret sharing using various classes of error-correcting codes [6, 7].

Organization. The paper is organized as follows. Section 2 contains definitions and results we use. In Section 3 we show that $t < n/2$ is necessary for secure computation of f_G. In Sections 4.2 and 4.3 we show how to construct a t-private protocol for f_G given a 't-Reliable' colouring of a planar graph. Then in Section 4.4, we present two constructions of such t-Reliable colourings. Finally, Section 4.5 summarizes some generalizations and extensions, and Section 5 concludes with some open problems.

2 Preliminaries

We recall the definition of secure multi-party computation in the passive (semi-honest), computationally unbounded attack model, restricted to deterministic symmetric functionalities and perfect emulation [13]. Let $[n]$ denote the set $\{1, \ldots, n\}$.

Definition 2.1 *Let $f : (\{0,1\}^*)^n \to \{0,1\}^*$ denote an n-input, single-output function, and let Π be an n-party protocol for computing f. We denote the party input sequence by $\mathbf{x} = (x_1, \ldots, x_n)$, the joint protocol view of parties in subset $I \subseteq [n]$ by $\mathsf{VIEW}_I^{\Pi}(\mathbf{x})$, and the protocol output by $\mathsf{OUT}^{\Pi}(\mathbf{x})$. For $0 < t < n$, we say that Π is a t-private protocol for computing f if there exists a probabilistic polynomial-time algorithm S, such that, for every $I \subset [n]$ with $\#I \leq t$ and every $\mathbf{x} \in (\{0,1\}^*)^n$, the random variables*

$$\langle \mathsf{S}(I, \mathbf{x}_I, f(\mathbf{x})), f(\mathbf{x}) \rangle \text{ and } \langle \mathsf{VIEW}_I^{\Pi}(\mathbf{x}), \mathsf{OUT}^{\Pi}(\mathbf{x}) \rangle$$

are identically distributed, where $\mathbf{x}_I$ denotes the projection of the n-ary sequence $\mathbf{x}$ on the coordinates in I.

To prove our result we will invoke a combinatorial characterization of 2-input functions for which a 1-private 2-party computation protocol exists, due to Kushilevitz [15]. To state this result, we need the following definitions.

Definition 2.2 *Let $M = C \times D$ be a matrix, where C is the set of rows and D is the set of columns. Define a binary relation $\sim$ on pairs of rows of M as follows: $x_1, x_2 \in C$ satisfy $x_1 \sim x_2$ if there exists $y \in D$ such that $M_{x_1,y} = M_{x_2,y}$. Let $\equiv$ denote the equivalence relation on the rows of M which is the transitive closure of $\sim$. Similarly, we define $\sim$ and $\equiv$ on the columns of M.*

Definition 2.3 *A matrix M is called* forbidden *if all its rows are equivalent, all its columns are equivalent, and not all entries of M are equal.*

Definition 2.4 *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0, \ldots, m-1\}$ be any 2-input function. A matrix M for f is a $2^n \times 2^n$ matrix with entries in $\{0, \ldots, m-1\}$, where each row x of f corresponds to a value for the first input to f, each column y corresponds to a value for the second input to f, and the entry $M_{x,y}$ contains the value $f(x, y)$.*

Theorem 2.1 (Kushilevitz [15]) *Let f be a 2-input function and let M be a matrix for f. Then a 1-private 2-party protocol for computing f exists if and only if M does not contain a forbidden submatrix.*

3 Honest Majority is Necessary for n-Product in Non-Abelian Groups

We show that an honest majority $t < n/2$ is necessary for secure computation of the n-product function in non-abelian groups.

Theorem 3.1 *Let $(G, \cdot)$ denote a finite non-abelian group and let $n \geq 4$. There does not exist a $\lceil \frac{n}{2} \rceil$-private protocol for computing $f_G(x_1, \ldots, x_n) = x_1 \cdot x_2 \cdots x_n$.*

Proof. The proof proceeds by contradiction; we show that if a $\lceil \frac{n}{2} \rceil$-private protocol Π exists for f_G for $n \geq 4$, then we can construct a 1-private 2-party protocol for a 2-input function f'_G whose matrix M' contains a forbidden submatrix, thus contradicting Theorem 2.1.

Lemma 3.1 *Suppose there exists a $\lceil \frac{n}{2} \rceil$-private n-party protocol Π for computing the n-input function $f_G : G^n \to G$ defined by $f_G(x_1, \ldots, x_n) = x_1 \cdots x_n$ for $n \geq 4$. Then we can construct a 1-private 2-party protocol Π' for computing the 2-input function $f'_G : G^2 \times G^2 \to G$ defined by $f'_G((x'_1, x'_3), (x'_2, x'_4)) = x'_1 \cdot x'_2 \cdot x'_3 \cdot x'_4$.*

Proof. Given party P'_1 with input (x'_1, x'_3) and party P'_2 with input (x'_2, x'_4), the protocol Π' runs as follows. First, if $n \geq 5$, we partition the set $\{5, \ldots, n\}$ into two disjoint subsets S'_1 and S'_2 such that the size of both S'_1 and S'_2 is at most $\lceil \frac{n}{2} \rceil - 2$ (namely, if n is even we take $\#S'_1 = \#S'_2 = n/2 - 2$, and if n is odd we take $\#S'_1 = (n-3)/2$ and $\#S'_2 = (n-5)/2$). Then $\Pi'(P'_1, P'_2)$ consists of running the n-party protocol $\Pi(P_1, \ldots, P_n)$ where:

- P'_1 plays the role of parties $(P_1, P_3, \{P_i\}_{i \in S'_1})$ in Π, and sets those parties inputs to be $x_1 = x'_1$, $x_3 = x'_3$, and $x_i = 1$ for all $i \in S'_1$, respectively.
- P'_2 plays the role of parties $(P_2, P_4, \{P_i\}_{i \in S'_2})$ in Π, and sets those parties inputs to be $x_2 = x'_2$, $x_4 = x'_4$ and $x_i = 1$ for all $i \in S'_2$, respectively.

The 1-privacy of protocol $\Pi'(P'_1, P'_2)$ for computing f'_G follows from the $\lceil \frac{n}{2} \rceil$-privacy of protocol $\Pi(P_1, \ldots, P_n)$ for computing f_G because:

- $f_G(x'_1, x'_2, x'_3, x'_4, 1, \ldots, 1) = f'_G(x'_1, x'_2, x'_3, x'_4) = x'_1 \cdot x'_2 \cdot x'_3 \cdot x'_4$ for all $x'_1, x'_2, x'_3, x'_4 \in G$.

- For each (x'_1, x'_2, x'_3, x'_4), the view of P'_1 (resp. P'_2) in protocol $\Pi'(P'_1, P'_2)$ is identical to the view of a set of at most $\lceil \frac{n}{2} \rceil$ parties in protocol $\Pi(P_1, \ldots, P_n)$ whose inputs are known to P'_1 (resp. P'_2), with special settings of 1 for some inputs. Thus the same view simulator algorithm S of Π can be used to simulate the view in Π'.

This completes the proof. □

Lemma 3.2 *For any non-abelian group G, the matrix M for the 2-input function $f'_G : G^2 \times G^2 \to G$ defined by $f'_G((x'_1, x'_3), (x'_2, x'_4)) = x'_1 \cdot x'_2 \cdot x'_3 \cdot x'_4$ contains a 2×2 forbidden submatrix.*

Proof. Observe from Definitions 2.2 and 2.3 that any 2×2 matrix with 3 equal elements and a fourth distinct element is a forbidden matrix. Now recall that the rows of matrix M for f'_G are indexed by $(x'_1, x'_3) \in G^2$, the columns of M are indexed by $(x'_2, x'_4) \in G^2$, and the entry of M at row (x'_1, x'_3) and column (x'_2, x'_4) is $M_{(x'_1,x'_3),(x'_2,x'_4)} = x'_1 \cdot x'_2 \cdot x'_3 \cdot x'_4$. Also, since G is non-abelian, there exist a pair of elements a and b in G such that a and b do not commute and $a, b \neq 1$. Consider the 2×2 submatrix of M formed by the intersections of the 2 rows $(1,1)$ and (a, a^{-1}) and the 2 columns $(1,1)$ and (b, b^{-1}) (these row and column pairs are distinct because $a, b \neq 1$). We claim that this submatrix is forbidden. Indeed, three of the submatrix entries are equal because $M_{(1,1),(1,1)} = M_{(a,a^{-1}),(1,1)} = M_{(1,1),(b,b^{-1})} = 1$, and the remaining fourth entry is distinct because $M_{(a,a^{-1}),(b,b^{-1})} = a \cdot b \cdot a^{-1} \cdot b^{-1} = (a \cdot b) \cdot (b \cdot a)^{-1} \neq 1$ since a and b do not commute. This completes the proof. □

Combining Lemma 3.1 and Lemma 3.2, we conclude that if a $\lceil \frac{n}{2} \rceil$-private protocol Π exists for f_G for $n \geq 4$, then we obtain a contradiction to Theorem 2.1. This completes the proof. □

4 Constructions

4.1 Our Approach: Black Box Non-Abelian Group Protocols

Our protocols will treat the group G as a black box in the sense that the only computations performed by players in our protocols will be one of the following three: Multiply (Given $x \in G$ and $y \in G$, compute $x \cdot y$), Inverse (Given $x \in G$, compute x^{-1}), and Random Sampling (Choose a uniformly random $x \in G$). It is easy to see that these three operations are sufficient for implementing a perfect k-of-k threshold secret sharing scheme. We use this k-of-k scheme as a fundamental building block in our protocols. The following proposition is easy to prove.

Proposition 4.1 *Fix $x \in G$ and integers k and $j \in [k]$, and suppose we create a k-of-k sharing $(s_x(1), s_x(2), \ldots, s_x(k))$ of x by picking the $k-1$ shares $\{s_x(i)\}_{i \in [k] \setminus \{j\}}$ uniformly and independently at random from G, and computing $s_x(j)$ to be the unique element of G such that $x = s_x(1)s_x(2) \cdots s_x(k)$. Then the distribution of the shares $(s_x(1), s_x(2), \ldots, s_x(k))$ is independent of j.*

4.2 Construction of n-Product Protocol from a Shared 2-Product Subprotocol

We begin by reducing the problem of constructing a t-private protocol for the n-product function $f(x_1, \ldots, x_n) = x_1 \cdots x_n$ (where party P_i holds input x_i for $i = 1, \ldots, n$), to the problem of constructing a subprotocol for the *Shared* 2-Product function $f'(x, y) = x \cdot y$, where inputs x, y and output $z = x \cdot y$ are shared among the parties. We define for this subprotocol a so-called *strong* t-privacy definition, which will be needed later to prove the (standard) t-privacy of the full n-product protocol built from subprotocol Π_S. The definition of strong t-privacy requires the adversary's view simulator to simulate *all* output shares except one share not held by the adversary, in addition to simulating the internal subprotocol view of the adversary.

Definition 4.1 (Shared n-Party 2-Product Subprotocol) *A n-Party Shared 2-Product subprotocol Π_S with sharing parameter ℓ and share ownership functions $\mathcal{O}_x, \mathcal{O}_y, \mathcal{O}_z : [\ell] \to [n]$ has the following features:*

- *Input: For $j = 1, \ldots, \ell$, party $P_{\mathcal{O}_x(j)}$ holds jth share $s_x(j) \in G$ of x and party $P_{\mathcal{O}_y(j)}$ holds jth share $s_y(j) \in G$ of y, where $\boldsymbol{s}_x = (s_x(1), s_x(2), \ldots, s_x(\ell))$ and $\boldsymbol{s}_y = (s_y(1), s_y(2), \ldots, s_y(\ell))$ denote ℓ-of-ℓ sharing of $x \stackrel{\text{def}}{=} s_x(1) \cdot s_x(2) \cdots s_x(\ell)$ and $y \stackrel{\text{def}}{=} s_y(1) \cdot s_y(2) \cdots s_y(\ell)$, respectively.*

- *Output: For $j = 1, \ldots, \ell$, party $P_{\mathcal{O}_z(j)}$ holds jth share $s_z(j)$ of output product $z \stackrel{\text{def}}{=} s_z(1) \cdots s_z(\ell)$.*

- *Correctness: We say that that Π_S is correct if, for all protocol inputs $\boldsymbol{s}_x = (s_x(1), s_x(2), \ldots, s_x(\ell))$ and $\boldsymbol{s}_y = (s_y(1), s_y(2), \ldots, s_y(\ell))$, the output shares $\boldsymbol{s}_z = (s_z(1), s_z(2), \ldots, s_z(\ell))$ satisfy*

$$z = x \cdot y$$

where $x \stackrel{\text{def}}{=} s_x(1) \cdot s_x(2) \cdots s_x(\ell)$, $y \stackrel{\text{def}}{=} s_y(1) \cdot s_y(2) \cdots s_y(\ell)$ and $z \stackrel{\text{def}}{=} s_z(1) \cdots s_z(\ell)$.

- *Strong t-Privacy: We say that Π_S achieves **strong t-privacy** if there exists a probabilistic simulator algorithm S_{Π_S} such that for all $I \subset [n]$ with $\#I \leq t$, there exist $j_x^*, j_y^*, j_z^* \in [\ell]$ with $j_z^* \in \{j_x^*, j_y^*\}$, $\mathcal{O}_x(j_x^*) \notin I$, $\mathcal{O}_y(j_y^*) \notin I$ and $\mathcal{O}_z(j_z^*) \notin I$, such that for all protocol inputs $\boldsymbol{s}_x = (s_x(1), \ldots, s_x(\ell))$ and $\boldsymbol{s}_y = (s_y(1), \ldots, s_y(\ell))$, the random variables*

$$\langle \mathsf{S}_{\Pi_S}(I, \{\boldsymbol{s}_x(j)\}_{j \in [\ell] \setminus \{j_x^*\}}, \{\boldsymbol{s}_y(j)\}_{j \in [\ell] \setminus \{j_y^*\}}) \rangle \text{ and}$$
$$\langle \mathsf{VIEW}_I^{\Pi_S}(\boldsymbol{s}_x, \boldsymbol{s}_y), \{s_z(j)\}_{j \in [\ell] \setminus \{j_z^*\}} \rangle$$

*are identically distributed (over the random coins of Π_S). Here $\mathsf{VIEW}_I^{\Pi_S}(\boldsymbol{s}_x, \boldsymbol{s}_y)$ denotes the view of I in subprotocol Π_S run with input shares $\boldsymbol{s}_x, \boldsymbol{s}_y$, and $s_z(j)$ denotes the jth output share. If $j_z^* = j_x^*$ (resp. $j_z^* = j_y^*$) then we say Π_S achieves x-**preserving strong t-privacy** (resp. y-**preserving strong t-privacy**). If $j_z^* = j_x^* = j_y^*$ for all I, then we say Π_S achieves **symmetric strong t-privacy**.*

Remark 1: The share ownership functions $\mathcal{O}_x, \mathcal{O}_y, \mathcal{O}_z$ specify for each share index $j \in [\ell]$, the indices $\mathcal{O}_x(j), \mathcal{O}_y(j), \mathcal{O}_z(j)$ in $[n]$ of the party which holds the jth input shares $s_x(j)$ and $s_y(j)$ and jth output share $s_z(j)$, respectively.
Remark 2: The adversary view simulator S_{Π_S} for collusion I is given all input shares except the j_x^*th x-share $s_x(j_x^*)$ and j_y^*th y-share $s_y(j_y^*)$ (where $j_x^*, j_y^* \in [\ell]$, which depend on I, are indices of shares given to players *not* in I), and outputs all output shares except the j_z^*th share $s_z(j^*)$ of z. The x-preserving strong t-privacy property ensures that, for each I, the same value of index $j_z^* = j_x^*$ is used for both x-input shares and output shares. This allows multiple simulator runs to be composed, using output shares of one subprotocol run as x-input shares in a following subprotocol run, as shown in the security proof of the following construction. If in addition, *symmetric* strong t-privacy is achieved, one can use output shares of one subprotocol run as either x-input *or* y-input shares for the following subprotocol run, allowing for more efficient protocols. Alternatively, instead of one subprotocol Π_S which achieves symmetric strong t-privacy, we may use a pair of subprotocols $\Pi_S^{(x)}$, $\Pi_S^{(y)}$ which are x-preserving and y-preserving, respectively, and are *compatible* in the following sense.

Definition 4.2 (Compatible Subprotocols) *Let $\Pi_S^{(x)}$ and $\Pi_S^{(y)}$ denote two shared 2-Product subprotocols which satisfy x-preserving strong t-privacy and y-preserving strong t-privacy, respectively. We say $\Pi_S^{(x)}$ and $\Pi_S^{(y)}$ are* compatible *if:*

- *$\Pi_S^{(x)}$ and $\Pi_S^{(y)}$ have the same share ownership functions $\mathcal{O}_x, \mathcal{O}_y$.*

- *For each collusion $I \subset [n]$ with $\#I \leq t$, $\Pi_S^{(x)}$ and $\Pi_S^{(y)}$ have the same j_x^* index and the same j_y^* index (defined as in Def. 4.1).*

The idea is that if $\Pi_S^{(x)}$ and $\Pi_S^{(y)}$ are compatible, we can use the output shares of a $\Pi_S^{(x)}$ run as x-input shares to a following $\Pi_S^{(y)}$ run, because the view simulator for the first subprotocol run simulates all output shares except the j_x^*th one (by the x-preserving property), while the view simulator for the second subprotocol run requires as input all x-input shares except the $\bar{j}_x^*$th one. Since $\bar{j}_x^* = j_x^*$ by compatability, we can use the output of the first simulator as input to the second simulator. Note that if Π_S satisfies symmetric strong t-privacy then Π_S is compatible with itself, i.e. one can take $\Pi_S^{(x)} = \Pi_S^{(y)} = \Pi_S$. Refer to Fig. 1(a) (resp. Fig. 1(b)) for an example of an x-preserving (resp. y-preserving) shared 2-product subprotocol which achieves strong 2-privacy.

We now explain our construction of an n-Product Protocol $\Pi(T, \Pi_S^{(x)}, \Pi_S^{(y)})$ given a binary computation tree T for f_G with n leaf nodes corresponding to the n protocol inputs (as illustrated in Fig. 1(c)), and a pair of compatible Shared 2-Product subprotocols $\Pi_S^{(x)}$, $\Pi_S^{(y)}$. We assume that $\Pi_S^{(x)}$ (resp. $\Pi_S^{(y)}$) satisfy x-preserving (resp. y-preserving) strong t-privacy, with sharing parameter ℓ and share ownership functions $\mathcal{O}_x, \mathcal{O}_y, \mathcal{O}_z$.

The protocol Π begins with each party P_j computing an ℓ-of-ℓ sharing of its input x_j, and distributing out these shares to the n parties according to the share ownership functions $\mathcal{O}_x, \mathcal{O}_y$. Then, for each internal node N of the tree T, protocol Π performs the 2-product computation associated with node N on ℓ-of-ℓ sharings of the values of node N's two children nodes. This is done by running one of the shared 2-product subprotocols $\Pi_S^{(x)}$ or $\Pi_S^{(y)}$, resulting in an ℓ-of-ℓ sharing of the internal node value. The choice of which subprotocol to run at node N is determined by N's relationship to its parent node: if N is a *left* child, subprotocol $\Pi_S^{(x)}$ is used, else if N is a *right* child, subprotocol $\Pi_S^{(y)}$ is used (if N is the root node, either subprotocol may be used). Eventually this recursive process gives an ℓ-of-ℓ sharing of the root node value $x_1 \cdots x_n$, which is broadcast to all parties. We refer the reader to Fig 1(d) for an example illustration of the composition of three subprotocol runs corresponding to three internal nodes in the tree in Fig. 1(c), using the subprotocols illustrated in Fig. 1(a)-(b). The Appendix A contains a formal specification of this construction.

Remark. As illustrated in the example in Fig. 1, the shares of a subprotocol input (say x-input shares) may be held by a subset of only $t+1$ of the n players, and some players in this subset may hold more than one input share. This is in contrast to classical protocols [4, 5], where subprotocol inputs are shared among all n players, with each player holding one input share.

The following Lemma establishes the t-privacy of protocol $\Pi(T, \Pi_S^{(x)}, \Pi_S^{(y)})$, assuming the correctness and strong t-privacy of subprotocols $\Pi_S^{(x)}$,$\Pi_S^{(y)}$. Refer to Appendix B for the proof.

Lemma 4.1 *For any binary tree T with n leaves, if the n-party compatible Shared 2-Product subprotocols $\Pi_S^{(x)}$ (resp. $\Pi_S^{(y)}$) satisfy correctness and x-preserving (resp. y-preserving) strong t-privacy (see Def. 4.1 and Def. 4.2), then protocol $\Pi(T, \Pi_S^{(x)}, \Pi_S^{(y)})$ is an n-party t-private protocol for computing n-Product function $f_G(x_1, \ldots, x_n) = x_1 \cdots x_n$.*

4.3 Construction of a t-Private n-Party Shared 2-Product Subprotocol from a t-Reliable n-Colouring of a Planar Graph

Next, we reduce the problem of constructing a t-Private n-Party Shared 2-Product Subprotocol Π_S to a combinatorial problem defined below of finding a 't-Reliable n-Colouring' of the nodes of a planar graph. We note that our notion of a 't-Reliable n-Colouring' is closely related to a similar notion defined in [10], and shown to be equivalent to the existence of private communication via a network graph in which each node is assigned one of n possible colours and the adversary controls all nodes with colours belonging to a t-colour subset I.

Consider a Planar Directed Acyclic Graph (PDAG) $\mathcal{G}$ having 2ℓ source (input) nodes drawn in a horizontal row at the top, ℓ sink (output) nodes drawn in a horizontal row at the bottom, and $\sigma_{\mathcal{G}}$ nodes overall. We use PDAG $\mathcal{G}$ to represent a blackbox protocol, where the input/output nodes are labelled by the protocol input/output group elements, and the internal graph nodes are labelled

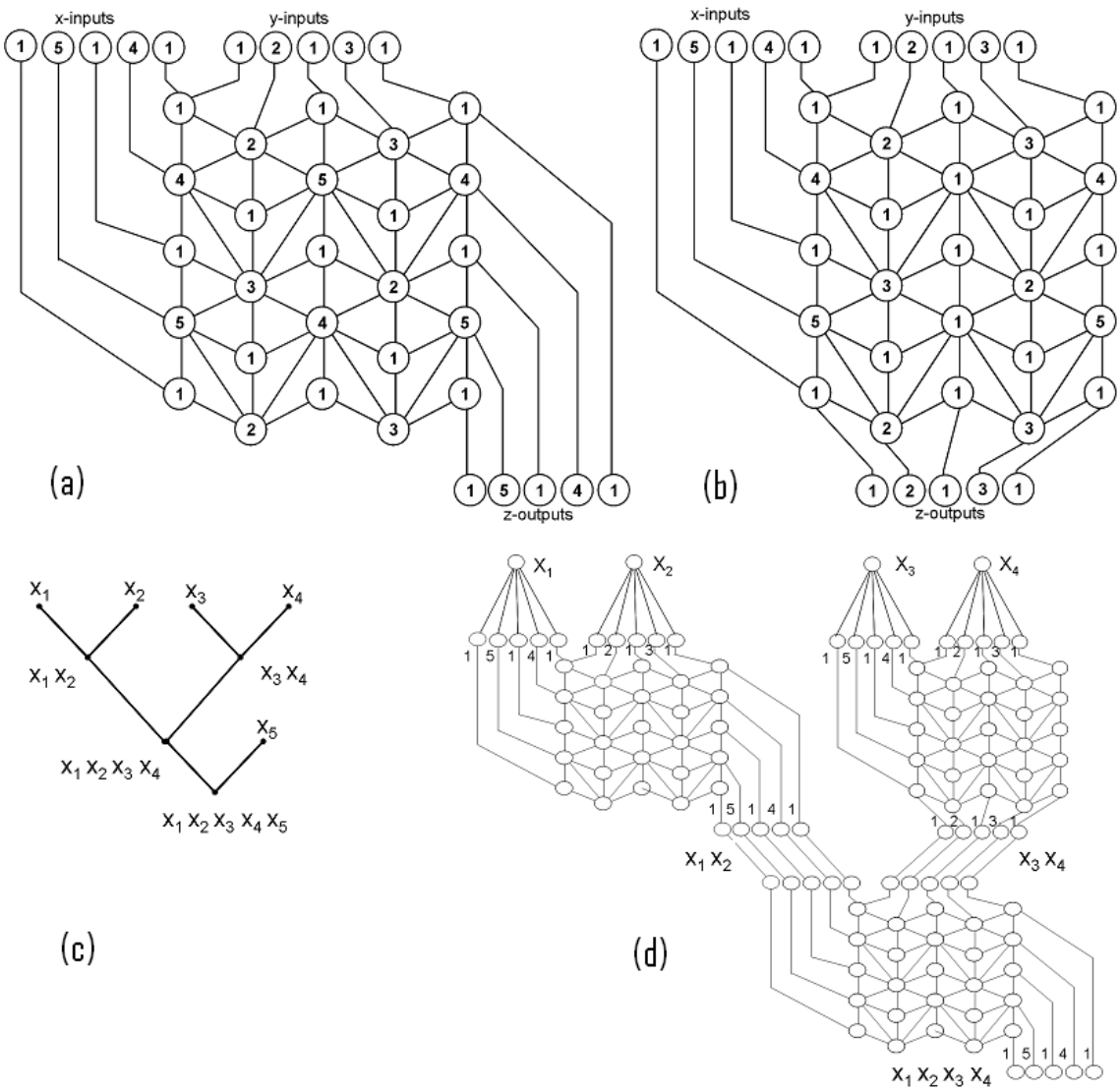

Figure 1: (a) Example of a 5-party shared 2-Product subprotocol $\Pi_S^{(x)}$ satisfying x-preserving strong 2-privacy, with sharing parameter $\ell = 5$. The source nodes on the top row are labeled with indices of parties holding x-input and y-input shares, according to share ownership functions $\mathcal{O}_x, \mathcal{O}_y$. The sink nodes on the bottom row are labeled with indices of parties holding the output shares according to share ownership function $\mathcal{O}_z$ (note that $\mathcal{O}_z = \mathcal{O}_x$). Communication is from top to bottom. At each internal node, the party whose index labels the internal node multiplies the shares received on incoming edges, and splits the result into shares which are sent along the outgoing edges (see Section 4.3 for more details). (b) Example of a 5-party shared 2-Product subprotocol $\Pi_S^{(y)}$ satisfying y-preserving strong 2-privacy. Note it is identical to $\Pi_S^{(x)}$, except that outputs are taken from a different set of nodes. Also note that $\mathcal{O}_z = \mathcal{O}_y$. (c) Example of a binary computation tree T for f_G with $n = 5$. Leaf nodes correspond to the inputs $x_1, \ldots, x_5$, and each internal node corresponds to a multiplication in G. (d) Illustration of three subprotocol runs (corresponding to the top 3 internal nodes in tree T in (c)) in the protocol $\Pi(T, \Pi_S^{(x)}, \Pi_S^{(y)})$ for f_G constructed from tree T in (c) and subprotocols $\Pi_S^{(x)}$,$\Pi_S^{(y)}$ in (a),(b).

by intermediate protocol values. Each internal graph node is also assigned a *colour* specifying the player which computes the internal node value. The graph edges represent group elements sent from one player to another. The computation performed at each node is multiplication of the values on all incoming edges and resharing the product along the outgoing edges using the k-of-k secret sharing scheme in Proposition 4.1. All computations in the ith round of the 2-Product subprotocol correspond to the ith row (from the top) in the PDAG. Communications between nodes correspond to edges between consecutive rows.

Actually to construct a protocol for any non-abelian group our requirement on graph $\mathcal{G}$ is slightly stronger than planarity and can be precisely defined as follows.

Definition 4.3 (Admissible PDAG) *We call graph $\mathcal{G}$ an* Admissible PDAG with share parameter ℓ and size parameter m *if it has the following properties:*

- *Nodes of $\mathcal{G}$ are drawn on a square $m \times m$ grid of points (each node of $\mathcal{G}$ is located at a grid point but some grid points may not be occupied by nodes). Rows of the grid are indexed from top to bottom and columns from left to right by the integers $1, 2, \ldots, m$. A node of $\mathcal{G}$ at row i and column j is said to have index (i, j). $\mathcal{G}$ has 2ℓ source (input) nodes on top row 1, and ℓ sink (output) nodes on bottom row m.*

- *Incoming edges of a node on row i only come from nodes on row $i-1$, and outgoing edges of a node on row i only go to nodes on row $i+1$.*

- *For each row i and column j, let $\eta_1^{(i,j)} < \ldots < \eta_{q^{(i,j)}}^{(i,j)}$ denote the ordered column indices of the $q^{(i,j)} > 0$ nodes on level $i+1$ which are connected to node (i, j) by an edge. Then, for each $j = 1, \ldots, m-1$, we have*
$$\eta_{q^{(i,j)}}^{(i,j)} \leq \eta_1^{(i,j+1)}, \tag{1}$$
i.e. the rightmost node on level $i+1$ connected to node (i, j) is to the left of (or equal to) the leftmost node on level $i+1$ connected to node $(i, j+1)$.

We call the left ℓ source nodes on row 1 (indexed $(1,1), \ldots, (1,\ell)$) the 'x-input' nodes and the last ℓ source nodes on row 1 (indexed $(1,\ell+1), \ldots, (1,2\ell)$) the '$y$-input' nodes. By ith x-input node, we mean the x-input node at position i from the left. We define the ith y-input and ith output node similarly.

Let $C : [m] \times [m] \to [n]$ be an n-Colouring function that associates to each node (i, j) of $\mathcal{G}$ a colour $C(i, j)$ chosen from a set of n possible colours $[n]$. We now define the notion of a t-Reliable n-Colouring.

Definition 4.4 (t-Reliable n-Colouring) *We say that $C : [m] \times [m] \to [n]$ is a t-Reliable n-Colouring for admissible PDAG $\mathcal{G}$ (with share parameter ℓ and size parameter m) if for each t-colour subset $I \subset [n]$, there exist $j_x^*, j_y^*, j_z^* \in [\ell]$ with $j_z^* \in \{j_x^*, j_y^*\}$ such that:*

- *There exists a path $PATH_x$ in $\mathcal{G}$ from the j_x^*th x-input node to the j_z^*th output node, such that none of the path node colours are in subset I (we call such a path I-avoiding), and*

- *There exists an I-avoiding path $PATH_y$ in $\mathcal{G}$ from the j_y^*th y-input node to the j_z^*th output node.*

If $j_z^* = j_x^*$ (resp. $j_z^* = j_y^*$) then we say that C is an x-preserving (resp. y-preserving) t-reliable n-Colouring. If $j_z^* = j_x^* = j_y^*$ for all I, then we say that C is a Symmetric *t-Reliable n-Colouring.*

Remark. The paths $PATH_x$ and $PATH_y$ in Definition 4.4 are free to move in *any* direction along each edge of directed graph $\mathcal{G}$, i.e. for this definition we regard $\mathcal{G}$ as an *undirected* graph (throughout the paper we assume that a path is *simple*, i.e. free of cycles; hence each node on the path is only visited once). An example of an admissible PDAG with I-avoiding paths $PATH_x$ and $PATH_y$ is

shown in Fig 2(a). Given an admissible PDAG $\mathcal{G}$ (with share parameter ℓ and size parameter m) and an associated t-Reliable n-Colouring $C : [m] \times [m] \to [n]$, we construct a t-Private n-Party Shared 2-Product Subprotocol $\Pi_S(\mathcal{G}, C)$.

Shared 2-Product Subprotocol $\Pi_S(\mathcal{G}, C)$

Input: We define the share ownership functions $\mathcal{O}_x, \mathcal{O}_y, \mathcal{O}_z$ of $\Pi_S(\mathcal{G}, C)$ according to the colours assigned by C to the input and output nodes of $\mathcal{G}$ (i.e. $\mathcal{O}_x(j) = C(1, j)$, $\mathcal{O}_y(j) = C(1, \ell + j)$, $\mathcal{O}_z(j) = C(m, j)$ for $j = 1, \ldots, \ell$). For $j = 1, \ldots, \ell$, party $P_{\mathcal{O}_x(j)}$ holds jth share $s_x(j) \in G$ of x and party $P_{\mathcal{O}_y(j)}$ holds jth share $s_y(j) \in G$ of y, where $\boldsymbol{s}_x = (s_x(1), s_x(2), \ldots, s_x(\ell))$ and $\boldsymbol{s}_y = (s_y(1), s_y(2), \ldots, s_y(\ell))$ denote ℓ-of-ℓ sharing of $x \stackrel{\text{def}}{=} s_x(1) \cdot s_x(2) \cdots s_x(\ell)$ and $y \stackrel{\text{def}}{=} s_y(1) \cdot s_y(2) \cdots s_y(\ell)$, respectively.

For each row $i = 1, \ldots, m$ and column $j = 1, \ldots, m$ of $\mathcal{G}$, party $P_{C(i,j)}$ does the following:

- $P_{C(i,j)}$ computes a label $v^{(i,j)}$ for node (i, j) of $\mathcal{G}$ as follows. If $i = 1$, $P_{C(i,j)}$ defines $v^{(i,j)} = s_x(j)$ for $j \leq \ell$ and $v^{(i,j)} = s_y(j)$ for $\ell + 1 \leq j \leq 2\ell$. If $i > 1$, $P_{C(i,j)}$ computes $v^{(i,j)}$ by multiplying the shares received from nodes at previous row $i - 1$ (labels of edges between a node on row $i - 1$ and node (i, j)), ordered from left to right according to the sender node column index.
- If $i = m$, $P_{C(m,j)}$ sets output share j to be the label $v^{(m,j)}$,
- else, if $i < m$, let $\eta_1^{(i,j)} < \ldots < \eta_{q^{(i,j)}}^{(i,j)}$ denote the ordered column indices of the nodes on level $i+1$ which are connected to node (i, j) by an edge. $P_{C(i,j)}$ chooses $q^{(i,j)} - 1$ uniformly random elements from G and computes a $q^{(i,j)}$-of-$q^{(i,j)}$ secret sharing $s_1^{(i,j)}, \ldots, s_{q^{(i,j)}}^{(i,j)}$ of label $v^{(i,j)}$ such that:

$$v^{(i,j)} = s_1^{(i,j)} \cdots s_{q^{(i,j)}}^{(i,j)}.$$

- For $k = 1, \ldots, q^{(i,j)}$, $P_{C(i,j)}$ sends share $s_k^{(i,j)}$ to party $P_{C(i+1,\eta_k^{(i,j)})}$ and labels edge from node (i, j) to node $(i + 1, \eta_k^{(i,j)})$ by the share $s_k^{(i,j)}$.

Note that the correctness of Π_S follows from the fact that the product of node values at each row of PDAG $\mathcal{G}$ is preserved and hence equal to $x \cdot y$, thanks to condition (1) in Definition 4.3.

Lemma 4.2 *If $\mathcal{G}$ is an admissible PDAG and C is an x-preserving (resp. y-preserving) t-Reliable n-Colouring for $\mathcal{G}$ then $\Pi_S(\mathcal{G}, C)$ achieves x-preserving (resp. y-preserving) strong t-privacy. Moreover, if C is a* Symmetric *t-Reliable n-Colouring, then $\Pi_S(\mathcal{G}, C)$ achieves* Symmetric *strong t-privacy.*

Proof. (Sketch) The full proof of Lemma 4.2 can be found in Appendix C. Here we only explain the main idea by considering the case when the I-avoiding paths $PATH_x$ and $PATH_y$ only have downward edges (in [9] we extend the argument to paths with upward edges). Consider $PATH_x$ from the j_z^*th x-input node to the j_x^*th output node. At the first node $PATH_x(1)$ on the path, although the node value $v(1) = s_x(j_x^*)$ is not known to the view simulator S_{Π_S}, we may assume, by Proposition 4.1, that in the real subprotocol Π_S, when node $PATH_x(1)$ shares out its node label among its q outgoing edges, it sends new random elements (labels) r_i on each of the $q - 1$ outgoing edges *not* on $PATH_x$. Thus simulator S_{Π_S} can easily simulate all outgoing edge values of $PATH_x(1)$ which are not on $PATH_x$. The same argument shows that for all kth nodes $PATH_x(k)$ and $PATH_y(k)$ on $PATH_x$ and $PATH_y$ respectively, simulator S_{Π_S} can simulate all values on outgoing edges of $PATH_x(k)$ and $PATH_y(k)$ which are *not* on $PATH_x$ or $PATH_y$ by independent random elements. The values on edges along $PATH_x$ or $PATH_y$ depend on the inputs $s_x(j_x^*)$ and $s_y(j_y^*)$ which are not known to simulator S_{Π_S}, but since the paths $PATH_x$ and $PATH_y$ are I-avoiding, these values are not in the view of I and need not be simulated by S_{Π_S}. Since S_{Π_S} knows all inputs to Π_S it can compute all other edge values in the Π_S, including all outputs except the j_z^*th one (which is on $PATH_x$ and $PATH_y$), as required. □

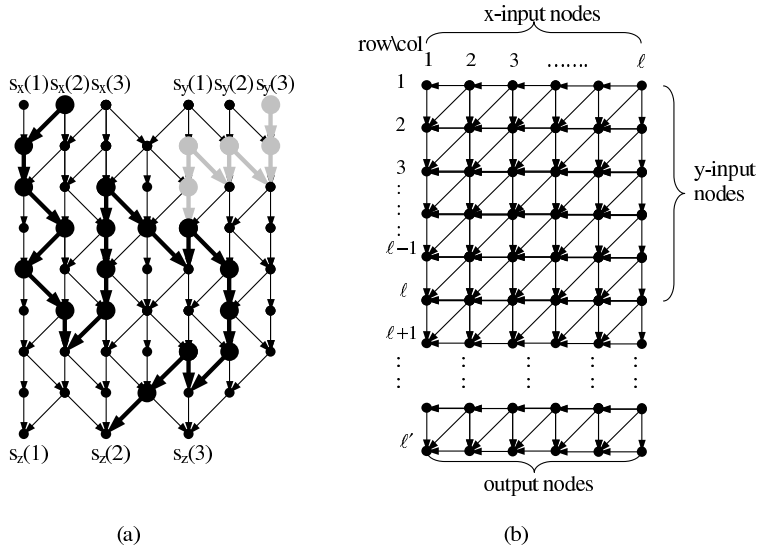

Figure 2: (a) Example of an admissible PDAG $\mathcal{G}$ with sharing parameter $\ell = 3$ (node colours are not indicated). For a given collusion I, an example I-avoiding path $PATH_x$ is shown in heavy black, and an example I-avoiding path $PATH_y$ (until the meeting with $PATH_x$) is shown in heavy gray. In this example, we have $j^* = 2$ and $j_y^* = 3$. (b) The admissible PDAG $\mathcal{G}_{tri}(\ell', \ell)$.

4.4 Constructions of t-Reliable n-Colourings of Planar Graphs

We now present two general constructions of t-Reliable n-Colourings of planar graphs which can be used to build t-Private n-Party protocols for the n-Product function in any finite group as explained in the previous sections. Our first deterministic construction achieves optimal collusion security ($t < n/2$) but has exponential complexity ($\ell = \binom{n}{t}$). Our second probabilistic construction has a slightly suboptimal collusion security ($t < n/2.948$) but has a very efficient linear complexity ($\ell = O(n)$).

The PDAG. The admissible PDAG $\mathcal{G}_{tri}(\ell', \ell)$ that we consider has sharing parameter ℓ and has $\ell' \times \ell$ nodes. It is shown in Fig. 2(b). The nodes of $\mathcal{G}_{tri}(\ell', \ell)$ are arranged in an $\ell' \times \ell$ node grid. Let (i, j) denote the node at row $i \in [\ell']$ (from the top) and column j (from the left). There are three types of edges in directed graph $\mathcal{G}_{tri}(\ell', \ell)$: (1) Horizontal edge: An edge connecting two adjacent nodes on the same row, directed from right to left (i.e. from node (i, j) to node $(i, j-1)$, for $i \in [\ell']$, $j \in [\ell] \setminus \{1\}$), (2) Vertical edge: An edge connecting two adjacent nodes on the same column, directed from top to bottom (i.e. from node (i, j) to node $(i+1, j)$, for $i \in [\ell'] \setminus \{\ell'\}$, $j \in [\ell]$), and (3) Diagonal edge: An edge connecting node (i, j) to node $(i+1, j-1)$, for $i \in [\ell'] \setminus \{\ell'\}$, $j \in [\ell] \setminus \{1\}$).

The ℓ nodes on the top row (row 1) of $\mathcal{G}_{tri}$ are the x-input nodes, indexed from left to right. The top ℓ nodes on the rightmost column of $\mathcal{G}_{tri}$ (column ℓ) are the y-input nodes, indexed from top to bottom.

Remark 1. The reader may notice that the above specification of $\mathcal{G}_{tri}$ does not formally satisfy the convention for drawing an admissible PDAG as defined in Def. 4.3, due to the horizontal edges and the fact that the y-input nodes are arranged along a column, rather than along the same row as the x-input nodes. However, it is easy to see that $\mathcal{G}_{tri}$ can also be drawn strictly according to Def. 4.3. Namely by rotating the drawing of $\mathcal{G}_{tri}$ in Fig. 2 by 45 degrees anticlockwise, the horizontal edges become diagonal edges, and x-inputs and y-inputs can be formally put on the same row by adding appropriate 'connecting' nodes of the same colour as the corresponding input nodes of $\mathcal{G}_{tri}$. These are only formal changes in drawing conventions, and there is no change in the protocol itself. In this section we use the drawing convention in Fig. 2 for clarity.

Remark 2. All diagonal edges in the definition of $\mathcal{G}_{tri}$ above are parallel (with a 'positive slope',

when using the drawing convention in Fig 2). However, it is clear that the admissible PDAG requirements are still satisfied if we remove from $\mathcal{G}_{tri}$ some 'positive slope' diagonal edges and add some 'negative slope' diagonal edges (connecting a node (i,j) to node $(i+1,j+1)$, for some $i \in [\ell'] \setminus \{\ell'\}$, $j \in [\ell] \setminus \{\ell\}$), as long as planarity of $\mathcal{G}$ is preserved (no two diagonal edges intersect). We denote such 'generalised' PDAGs by $\mathcal{G}_{gtri}$.

First Construction C_{comb} ($t < n/2$ and $\ell = \binom{n}{t}$). We now present an explicit construction of a t-Reliable n-Colouring C_{comb} of the square graph $\mathcal{G}_{tri}(\ell,\ell)$. The construction applies for all $n \geq 2t+1$ (i.e. $t \leq \lfloor \frac{n-1}{2} \rfloor$), and hence (by Section 3) the n-Product protocol constructed from it by the method of Sections 4.2 and 4.3 achieves $\lfloor \frac{n-1}{2} \rfloor$-privacy (which is optimal, as shown in Section 3). Unfortunately, the sharing parameter in this construction $\ell = \binom{n}{t}$, is exponential in t (and therefore the protocol communication cost is also exponential in t).

Colouring C_{comb} for graph $\mathcal{G}_{tri}(\ell,\ell)$ with $\ell = \binom{n}{t}$ and $n \geq 2t+1$

1. Let $I_1, \ldots, I_\ell$ denote the sequence of all $\ell = \binom{n}{t}$ t-colour subsets of $[n]$ (in some ordering).
2. For each $(i,j) \in [\ell] \times [\ell]$, define the colour $C(i,j)$ of node (i,j) of $\mathcal{G}_{tri}(\ell,\ell)$ to be any colour in the set $S_{i,j} = [n] \setminus (I_i \bigcup I_j)$ (note that since $|I_i| = |I_j| = t$ and $n \geq 2t+1$, the set $S_{i,j}$ contains at least $n - (|I_i| + |I_j|) \geq n - 2t \geq 1$ colours, so $S_{i,j}$ is never empty).

Lemma 4.3 *For $n \geq 2t+1$, the colouring C_{comb} is a Symmetric t-Reliable n-Colouring for graph $\mathcal{G}_{tri}(\ell,\ell)$, with $\ell = \binom{n}{t}$.*

Proof. Given each t-colour subset $I \subseteq [n]$, let j^* denote the index of I in the sequence $I_1, \ldots, I_\ell$ of all t-colour subsets used to construct C_{comb}, i.e $I_{j^*} = I$. By construction of C_{comb}, none of the nodes of $\mathcal{G}_{tri}(\ell,\ell)$ along column j^* have colours in $I_{j^*} = I$. Hence one can take column j^* of $\mathcal{G}_{tri}(\ell,\ell)$ as $PATH_x$. Similarly, we also know that none of the nodes of $\mathcal{G}_{tri}(\ell,\ell)$ along row j^* have colours in $I_{j^*} = I$, so one can take $PATH_y$ to consist of all nodes on row j^* which are on columns $j \geq j^*$, followed by all nodes on column j^* which are on rows $i \geq j^*$. Thus C_{comb} is a Symmetric t-Reliable n-Colouring for graph $\mathcal{G}_{tri}(\ell,\ell)$, as required. □

Remark. The colouring C_{comb} remains a Symmetric t-Reliable n-Colouring even if we remove all diagonal edges from $\mathcal{G}_{tri}(\ell,\ell)$ (since the paths $PATH_x$ and $PATH_y$ only contain vertical and horizontal edges).

Combining Lemma 4.3 (applied to a subset of $n' = 2t+1 \leq n$ colours from $[n]$) with Lemmas 4.1 and 4.2, we have

Corollary 4.1 *For any $t < n/2$, there exists a black-box t-private protocol for f_G with communication complexity $O(n\binom{2t+1}{t}^2)$ group elements.*

Second Construction C_{rand} ($t < n/2.948$ and $\ell = O(n)$). It is natural to ask whether the exponentially large sharing parameter $\ell = \binom{n}{t}$ can be reduced. Our second construction C_{rand} shows that this is certainly the case when $t < n/2.948$, achieving a linear sharing parameter $\ell = O(n)$.

As a first step towards our second construction, we relax the properties required from C in Definition 4.4 to slightly simpler requirements for the square graph $\mathcal{G}_{tri}(\ell,\ell)$ (i.e. $\ell' = \ell$), as follows.

Definition 4.5 (Weakly t-Reliable n-Colouring) *We say that $C : [\ell] \times [\ell] \to [n]$ is a* Weakly *t-Reliable n-Colouring for graph $\mathcal{G}_{tri}(\ell,\ell)$ if for each t-colour subset $I \subset [n]$:*

- *There exists an I-avoiding path P_x in $\mathcal{G}$ from a node on the top row (row 1) to a node on the bottom row (row ℓ). We call such a path an I-avoiding* top-bottom *path.*
- *There exists an I-avoiding path P_y in $\mathcal{G}$ from a node on the rightmost column (column ℓ) to a node on the leftmost column (column 1). We call such a path an I-avoiding* right-left *path.*

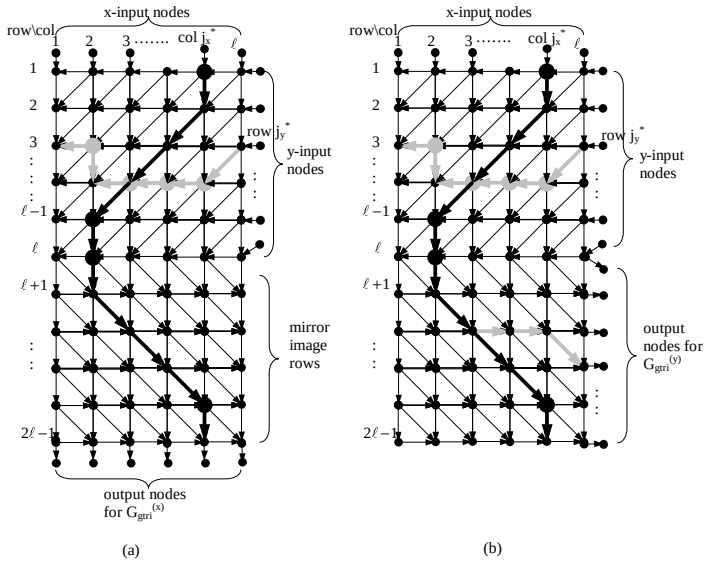

Figure 3: (a) Graph $\mathcal{G}_{gtri}^{(x)}(2\ell-1,\ell)$. (b) Graph $\mathcal{G}_{gtri}^{(y)}(2\ell-1,\ell)$.
Note that the colour of the input and output nodes is identical to the colour of the nodes connected to them.

Note that in the above definition of Weak t-Reliability, the index of the starting node of path P_x in the top row need not be the same as the index of the exit node of P_x in the bottom row (whereas in the definition of t-Reliability, $PATH_x$ must exit at the same position along the output row as the position in the top row where $PATH_x$ begins).

The following lemma shows that finding a Weakly t-Reliable n-Colouring for the square graph $\mathcal{G}_{tri}(\ell,\ell)$ is sufficient for constructing a (standard) t-Reliable n-Colouring for a rectangular graph $\mathcal{G}_{gtri}(2\ell-1,\ell)$. The idea is to add $\ell-1$ additional rows to $\mathcal{G}_{tri}(\ell,\ell)$ by appending a 'mirror image' (reflected about the last row) of itself, as shown in Fig. 3. Note that we define two versions of $\mathcal{G}_{gtri}(2\ell-1,\ell)$ called $\mathcal{G}_{gtri}^{(x)}(2\ell-1,\ell)$ and $\mathcal{G}_{gtri}^{(y)}(2\ell-1,\ell)$. The difference between them is only in the choice of output nodes: the bottom row ℓ nodes are used as output nodes for $\mathcal{G}_{gtri}^{(x)}(2\ell-1,\ell)$ as shown in Fig. 3(a) (giving an x-preserving colouring) and the last ℓ nodes on the rightmost columnn are used as output nodes for $\mathcal{G}_{gtri}^{(y)}(2\ell-1,\ell)$ as shown in Fig. 3(b) (giving a y-preserving colouring).

Lemma 4.4 *Let $C:[\ell]\times[\ell]\to[n]$ be a Weakly t-Reliable n-Colouring (see Def. 4.5) for square admissible PDAG $\mathcal{G}_{tri}(\ell,\ell)$. Then we can construct an x-preserving (resp. y-preserving) t-Reliable n-Colouring (see Def. 4.4) for a rectangular admissible PDAG $\mathcal{G}_{gtri}^{(x)}(2\ell-1,\ell)$ (resp. $\mathcal{G}_{gtri}^{(y)}(2\ell-1,\ell)$). Moreover, the 2-product subprotocols $\Pi_S^{(x)}$, $\Pi_S^{(y)}$ constructed from $\mathcal{G}_{gtri}^{(x)}(2\ell-1,\ell)$ and $\mathcal{G}_{gtri}^{(y)}(2\ell-1,\ell)$ (using Lemma 4.2) are compatible (see Def. 4.2).*

Proof. For each t-colour subset I, let P_x and P_y denote the top-bottom and right-left I-avoiding paths in $\mathcal{G}_{tri}(\ell,\ell)$ for the given colouring C. By planarity, it is clear that P_x and P_y must cross over (intersect) on at least one node. Let n^* denote the first node on P_y which is also on P_x (see Fig. 4(a)). We can modify right-left path P_y into a path P_y' that follows P_y until reaching the cross over node n^*, and then follows P_x to the exit. Hence we obtain a path P_y' that begins at a node on the rightmost column of $\mathcal{G}_{tri}(\ell,\ell)$ and exits at a node on the bottom row, namely the same bottom row node index j^* where P_x exits. We let j_x^* denote the index of the top row node where P_x begins (it is possible that $j_x^*\neq j^*$).

We construct a rectangular admissible PDAG $\mathcal{G}_{gtri}^{(x)}(2\ell-1,\ell)$ as follows. The first ℓ rows of $\mathcal{G}_{gtri}^{(x)}(2\ell-1,\ell)$ and interconnecting edges are identical to $\mathcal{G}_{tri}(\ell,\ell)$. For the bottom $\ell-1$ rows of $\mathcal{G}_{gtri}^{(x)}(2\ell-1,\ell)$, we take the 'mirror image' reflection of the top ℓ rows and their interconnecting edges,

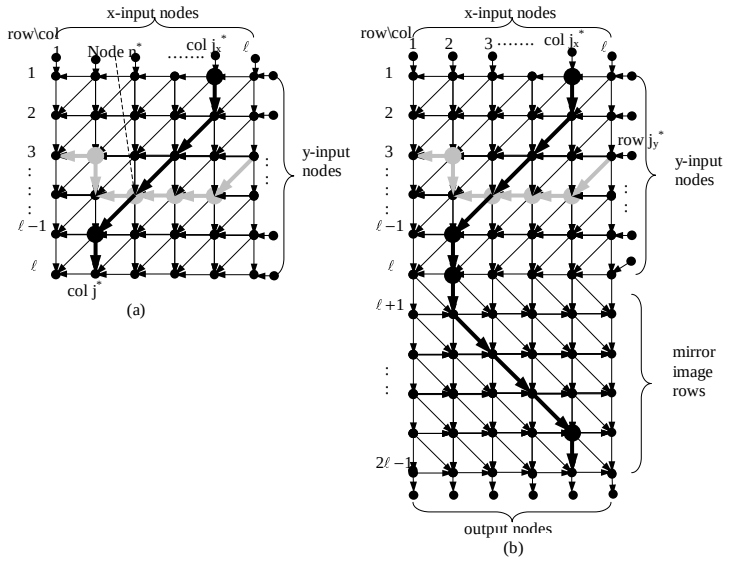

Figure 4: (a) Example paths in square PDAG $\mathcal{G}_{tri}(\ell,\ell)$ for a given Weakly t-Reliable n-Colouring (P_x in heavy black, P_y in heavy gray). (b) Corresponding paths in rectangular PDAG $\mathcal{G}_{gtri}(2\ell-1,\ell)$.

reflected about the ℓth row. The outputs are taken from the bottom ℓ nodes (see Fig. 4(b)). Note that each 'positive slope' diagonal edge between two rows among the first ℓ, give rise to 'negative slope' diagonal edges between two rows among the last ℓ. We construct the n-Colouring $C' : [2\ell-1]\times[\ell] \to [n]$ of $\mathcal{G}^{(x)}_{gtri}(2\ell-1,\ell)$ similarly, i.e. the top ℓ rows of $\mathcal{G}^{(x)}_{gtri}(2\ell-1,\ell)$ are coloured using C, i.e. $C'(i,j) = C(i,j)$ for $(i,j) \in [\ell]\times[\ell]$, and the last $\ell-1$ rows of $\mathcal{G}^{(x)}_{gtri}(2\ell-1,\ell)$ are coloured by a 'mirror image' of the first $\ell-1$ rows, i.e. $C'(\ell+i,j) = C(\ell-i,j)$ for $i \in [\ell-1]$ and $j \in [\ell]$.

We claim that C' is an x-preserving t-Reliable n-Colouring for $\mathcal{G}^{(x)}_{gtri}(2\ell-1,\ell)$. Indeed, thanks to the 'mirror symmetry' of C' and $\mathcal{G}^{(x)}_{gtri}(2\ell-1,\ell)$ about the ℓth row, the I-avoiding path P_x from node $(1,j^*_x)$ to node (ℓ,j^*) can be extended by its 'mirror image' to an I-avoiding path $PATH_x$ that exits at output node $(2\ell-1,j^*_x)$ of $\mathcal{G}^{(x)}_{gtri}(2\ell-1,\ell)$. Since it also reaches node (ℓ,j^*), the I-avoiding path P'_y evidently can also be extended along the same 'mirror image' path to an I-avoiding path $PATH_y$ that exits at output node $(2\ell-1,j^*_x)$, as shown in Fig. 3(b). Thus C' satisfies the requirements of an x-preserving t-Reliable n-Colouring for $\mathcal{G}^{(x)}_{gtri}(2\ell-1,\ell)$.

The PDAG $\mathcal{G}^{(y)}_{gtri}(2\ell-1,\ell)$ is identical to $\mathcal{G}^{(y)}_{gtri}(2\ell-1,\ell)$ except that the outputs are taken from the ℓ bottom nodes of the last column as shown in Fig. 4(b). It is easy to see by adapting the above argument (with $PATH_x$ and $PATH_y$ now leaving at the mirror image of the j^*_yth node, i.e. output node $(2\ell-j^*_y,\ell)$ of $\mathcal{G}^{(y)}_{gtri}(2\ell-1,\ell)$) that C' is a y-preserving t-Reliable n-Colouring for $\mathcal{G}^{(y)}_{gtri}(2\ell-1,\ell)$. Moreover, since for all collusions $PATH_x$ and $PATH_y$ are identical in the top half of $\mathcal{G}^{(y)}_{gtri}(2\ell-1,\ell)$ and $\mathcal{G}^{(x)}_{gtri}(2\ell-1,\ell)$, we see that j^*_x and j^*_y are the same in the two graphs so the corresponding subprotocols $\Pi^{(x)}_S$ and $\Pi^{(y)}_S$ are compatible, as claimed. □

For our second colouring construction, we use the 'probabilistic method' [1], namely we choose the colour of each node in the square graph $\mathcal{G}_{tri}(\ell,\ell)$ independently and uniformly at random from $[n]$. Although there is a finite error probability p that such a random n-Colouring will not be Weakly t-Reliable, we show that if $n/t > 2.948$ and we use a sufficiently large (but only *linear* in n) sharing parameter $\ell = O(n)$, then the error probability p can be made arbitrarily small. Moreover, p decreases exponentially fast with ℓ, so p can be easily made negligible.

Remark. Although our results allow one to efficiently (using $\ell = O(n\log(\delta^{-1}))$) generate colourings for $\mathcal{G}_{tri}(\ell,\ell)$ which are Weakly t-Reliable except for a negligible error probability $p \le \delta$, it is natural to ask whether one can efficiently generate colourings which we are *certain* to be Weakly

t-Reliable. In this connection, we note that for any *given* t-collusion I, and a colouring C, there exists an efficient algorithm (with run time linear in the number of nodes ℓ^2) to verify that the coloured graph contains I-avoiding top-bottom/right-left paths P_x and P_y. However, the naive approach to verify that C is Weakly t-Reliable requires running this algorithm $\binom{n}{t}$ times (once for every possible I), giving a run time exponential in t. We do not know whether there is an efficient algorithm to verify that a given a given colouring C of $\mathcal{G}_{tri}(\ell, \ell)$ is Weakly t-Reliable.

Colouring C_{rand} for graph $\mathcal{G}_{tri}(\ell, \ell)$ with $\ell = O(n)$ and $n \geq 2.948t$

For each $(i, j) \in [\ell] \times [\ell]$, choose the colour $C(i, j)$ of node (i, j) of $\mathcal{G}_{tri}(\ell, \ell)$ independently and uniformly at random from $[n]$.

To analyse this construction, we will make use of the following counting Lemma. Here, for any right-left path in $\mathcal{G}_{tri}(\ell, \ell)$, we define its *length* as the number of nodes on the path. We say a path is *minimal* if removing any node from the path disconnects the path.

Lemma 4.5 *The number $N_P(k, \ell)$ of minimal right-left paths of length k in graph $\mathcal{G}_{tri}(\ell, \ell)$ is upper bounded as*

$$N_P(k, \ell) \leq c(\mu) \cdot \ell \cdot \mu^k,$$

for some constants $\mu, c(\mu)$, with $\mu \leq 2.948$. We call the minimal possible value for μ the connective constant *of $\mathcal{G}_{tri}(\ell, \ell)$.*

Proof. For a minimal right-left path, there are ℓ possible starting nodes on the rightmost column. We may assume without loss of generality that the first edge of the path is not a vertical edge. For the ith starting node on the rightmost column, there are at most 2 possibilities for the first path edge: a horizontal edge, or a diagonal edge. For $j \geq 1$, let $N_i(j)$ denote the number of minimal paths in $\mathcal{G}_{tri}(\ell, \ell)$ of length j starting at the ith node on the rightmost column. Note that the paths counted in $N_i(j)$ are not necessarily right-left paths, i.e. the last node in the path need not be on the leftmost column.

We use induction on j to show $N_i(j) \leq 3^{j-1}$ for $j \geq 2$. We have already shown above the basis step $N_i(2) = 2 < 3$. For the induction step, suppose that $N_i(j) \leq 3^{j-1}$ for some $j \geq 2$. We show that $N_i(j+1) \leq 3^j$.

Consider each path P of length j. We claim that there are at most 3 possible choices for adding a $(j+1)$th node $P(j+1)$ to P to create a minimal path P' of length $j+1$. Let $P(j-1)$ and $P(j)$ denote the $(j-1)$th node and jth node of P, respectively.

Suppose first that $P(j)$ is is a boundary node of $\mathcal{G}_{tri}(\ell, \ell)$ (i.e. it is on row 1 or row ℓ or column 1 or column ℓ). Then $P(j)$ has degree at most 4, and one of the 4 nodes adjacent to $P(j)$ is $P(j-1)$, so there are at most 3 possible choices for $P(j+1)$, as required.

Now suppose that $P(j)$ is an internal node of $\mathcal{G}_{tri}(\ell, \ell)$. Then $P(j)$ has degree 6, and one of the 6 nodes adjacent to $P(j)$ is $P(j-1)$. Hence there are at most 5 possibilities for $P(j+1)$. But it is easy to verify that 2 of those 5 adjacent nodes of $P(j)$ must also be adjacent to $P(j-1)$. Hence, neither of these 2 nodes can be chosen as $P(j+1)$ since the resulting path P' will not be minimal (indeed, if $P(j+1)$ is chosen adjacent to $P(j-1)$ then internal node $P(j)$ could be removed from P' without disconnecting it). So there are at most 3 possibilities for $P(j+1)$ to keep P' minimal.

We conclude that any minimal path P of length j can be extended in at most 3 ways to a minimal path P' of length $j+1$. It follows that $N_i(j+1) \leq 3N_i(j) \leq 3^j$, which completes the inductive step. Since there are ℓ possible starting nodes on the rightmost column, we get $N_P(k, \ell) \leq \ell \cdot 3^k$, which proves $\mu \leq 3$.

We now show how to improve the connective constant upper bound to $\mu \leq 2.948$. This improvement is based on the fact that the bound $\mu \leq 3$ only takes into account a '1 edge history' of the path

to restrict the number of possible 'next' nodes by ruling out those which destroy the path minimality due to 3 node cycles. By taking into account m-edge history for larger $m > 1$, we can improve the bound by also ruling out m'-cycles for $m' > 3$. Here we examine the case of $m = 4$ edge history, ruling out $m' = 6$ node cycles, as well as $m' = 3$ node cycles (see [9] for some results with even larger m).

Consider the 6 node cycle C_6 in graph $\mathcal{G}_{tri}(\ell, \ell)$ shown in Fig. 5(a). For any minimal path P of length $j \geq 4$ whose last 4 edges match a sequence of 4 successive edges along C_6 (in either clockwise or anticlockwise sense, such as the 4 edges between nodes $P(j-4), P(j-3), P(j-2), P(j-1), P(j)$ in Fig. 5(a)), we have at most 2 possibilities (labelled n_1, n_2 in Fig. 5(a)) for choosing a $(j+1)$th node $P(j+1)$ to extend P to a minimal path P' of length $j+1$. This is because by minimality, only 3 possiblities are allowed for $P(j+1)$ to rule out 3-node cycles in P' (as shown above), and out of those 3 nodes, one (labelled n^* in Fig 5(a)) can be eliminated to rule out the 6-cycle C_6 from being contained in P'. This reduction from 3 to 2 possibilities for $P(j+1)$ when the last 4 edges of P match a sequence from C_6 will give us the improved upper bound on μ.

To analyse this improvement, let $S(j)$ denote the set of all minimal paths P in $\mathcal{G}_{tri}(\ell, \ell)$ of length j starting at the ith node on the rightmost column of $\mathcal{G}_{tri}(\ell, \ell)$. We partition $S(j)$ into 4 disjoint subsets $S_1(j), \ldots, S_4(j)$ according to the number of matches of the 4 last edges of P with a sequence of successive edges on C_6, namely:

- $S_4(j)$ denotes the subset of paths in $S(j)$ whose 4 last edges match a sequence of 4 successive edges along C_6 (in either clockwise or anticlockwise sense).
- For $k = 3, 2, 1$, $S_k(j)$ denotes the subset of paths in $S(j)$ which are not in $S_{k+1}(j)$, but whose k last edges match a sequence of k successive edges along C_6 (in either clockwise or anticlockwise sense).

For $j \geq 5$ and $k \in \{1, 2, 3, 4\}$, we say that a minimal path P of length j is in state k if $P \in S_k(j)$. We can now construct a finite state machine M whose state transition function specifies for each minimal path P of length j in state k, the possible 'next' state k' of a minimal path P' of length $j+1$ formed by adding a $(j+1)$th node to P. The state transition diagram of M is shown in Fig 5(b), where a label b on a transition from state k to k' indicates that there are b possibilities for the $(j+1)$th node which lead to this state transition. For example, as shown in Fig 5(a), if P is in state 4, then there are 2 possibilities for node $P(j+1)$: one (node labelled n_1) leads to a transition to state 1 (since no two successive edges in C_6 are in the same column), the other (node labelled n_2) leads to a transition to state 2 (since no three successive edges in C_6 are in the order 'horizontal, vertical, horizontal'). It is easy to verify that the same transition rule from state 4 holds for all paths P in state 4 (i.e. regardless of the particular sequence of 4 successive edges along C_6 which form the last 4 edges of P). The transition rules for the other three states are also easy to verify.

For $j \geq 5$ and $k \in \{1, 2, 3, 4\}$ let $N_k(j)$ denote the number of minimal paths (starting at ith node of the rightmost column of $\mathcal{G}_{tri}(\ell, \ell)$) of length j in state k. From the labelled state transition diagram of M in Fig 5(b), we immediately obtain the following recursive bound:

$$\begin{pmatrix} N_1(j+1) \\ N_2(j+1) \\ N_3(j+1) \\ N_4(j+1) \end{pmatrix} \leq A_M \cdot \begin{pmatrix} N_1(j) \\ N_2(j) \\ N_3(j) \\ N_4(j) \end{pmatrix}, \text{ where } A_M = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2}$$

It follows from (2) that the vector $\boldsymbol{N}(j) \stackrel{\text{def}}{=} [N_1(j)\ N_2(j)\ N_3(j)\ N_4(j)]^T$ satisfies

$$\boldsymbol{N}(j) \leq A_M^{j-5} \boldsymbol{N}(5) \tag{3}$$

for $j \geq 5$. The matrix A_M can be diagonalised into the form $A_M = Q \cdot D \cdot Q^{-1}$, where Q is a 4×4 invertible matrix having the eigenvectors of A_M as its columns, and D is a 4×4 diagonal matrix

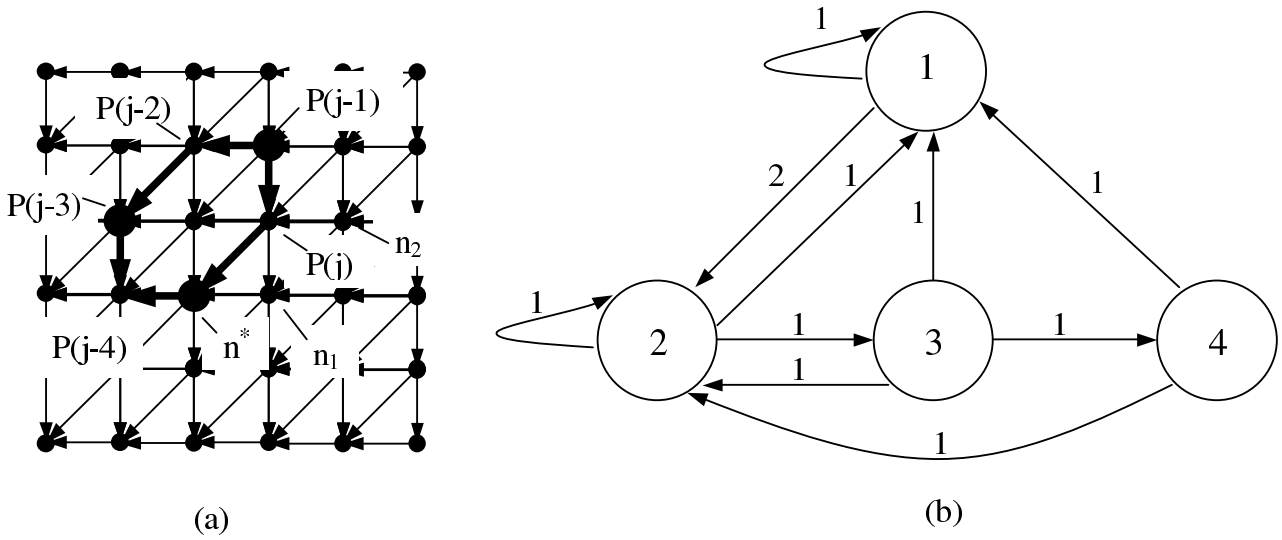

Figure 5: (a) The 6 node cycle C_6 in $\mathcal{G}_{tri}(\ell,\ell)$ is shown in heavy black. (b) The state transition diagram of finite state machine M.

having the 4 eigenvalues $\lambda_1,\ldots,\lambda_4$ of A_M on the diagonal. Note that $A_M^{j-5} = Q\cdot D^{j-5}\cdot Q^{-1}$, and D^{j-5} is a diagonal matrix with diagonal elements λ_k^{j-5} for $k=1,\ldots,4$. Plugging into (3) and adding up the components of $\boldsymbol{N}(j)$, we get the following upper bound on the number $N_P(j) = N_1(j)+\cdots N_4(j)$ of minimal paths of length j, starting at the ith node in the rightmost column of $\mathcal{G}_{tri}(\ell,\ell)$:

$$N_P(j) \le c_1\lambda_1^{j-5} + c_2\lambda_2^{j-5} + c_3\lambda_3^{j-5} + c_4\lambda_4^{j-5}, \tag{4}$$

where the constants $c_1,\ldots,c_4$ are determined from (3) by $\boldsymbol{N}(5)$ and the eigenvector matrix Q. It follows that $N_P(j) = O(\lambda^j)$, where $\lambda \stackrel{\text{def}}{=} \max_k|\lambda_k|$ is the largest eigenvalue magnitude of A_M. Numerical computation shows that $\lambda \le 2.948$, and hence (considering the ℓ possible starting nodes on the rightmost column of $\mathcal{G}_{tri}(\ell,\ell)$), the claimed bound $N_P(k,\ell) \le c(\mu)\cdot\ell\cdot\mu^k$ with $\mu=\lambda\le 2.948$ follows, for some constant $c(\mu)$. □

Remark 1. Our terminology *connective constant* for μ comes from similar (although not identical) constants defined in combinatorial studies of the 'self avoiding walk' in a lattice [17, 19]. However, the particular connective constant μ which arises in our work seems to not have been previously studied.

Remark 2. We have done some preliminary numerical eigenvalue computations using MATLAB with larger values of the 'edge history' parameter m on the path, extending our method for proving Lemma 4.5 (refer to [9] for more details). Using $m=8$ we obtained the improved bound $\mu\le 2.913$, although we are not yet certain about the accuracy of these MATLAB computations. We believe the efficient techniques from [17, 19] can be useful to further improve our numerical computed upper bound on μ by using even larger values of the 'edge history' on the path. Also, our method of bounding μ does not take into account the restriction that the paths of length k are right-left paths, so further improvements might result by taking this restriction into account.

Now we are ready to prove the following result.

Theorem 4.1 *Let $\mu, c(\mu)$ denote the connective constants of $\mathcal{G}_{tri}(\ell,\ell)$ (see Lemma 4.5). For any real constant $R>\mu$, if $t\le n/R$, there exists a Weakly t-Reliable n-Colouring for graph $\mathcal{G}_{tri}(\ell,\ell)$ for some $\ell = O(n)$. Moreover, for any constant $\delta>0$, the probability p that the random n-Colouring C_{rand} is not Weakly t-Reliable is upper bounded by δ if we choose*

$$\ell \ge b\cdot\frac{\log\binom{n}{t}}{\log(R/\mu)},$$

for a constant b satisfying

$$b - \left(\frac{3}{\log R}\right)\log b \geq 1 + \frac{\log\left(2c(\mu)\delta^{-1}\left(\frac{\log R}{\log(R/\mu)}\right)^3\right)}{\log R}. \tag{5}$$

Proof. Fix a t-colour subset I. We upper bound the probability $p(I)$, that if all ℓ^2 node colours of $\mathcal{G}_{tri}(\ell,\ell)$ are chosen uniformly and independently at random from $[n]$, the colouring C_{rand} is not Weakly t-Reliable, i.e. either an I-avoiding top-bottom path P_x doesn't exist, or an I-avoiding right-left path P_y doesn't exist.

Suppose that for a given colouring C, an I-avoiding top-bottom path P_x doesn't exist. This implies that the set $S(C)$ of graph nodes with colours in I must form a *top-bottom cutset*, which is defined as follows.

Definition 4.6 (Cutset/Minimal Cutset) *A set of nodes S in $\mathcal{G}_{tri}(\ell,\ell)$ is called a* top-bottom cutset *(resp.* right-left cutset*) if all top-bottom paths (resp. right-left paths) in $\mathcal{G}_{tri}(\ell,\ell)$ pass via a node in S. A cutset S is called* minimal *if removing any node from S destroys the cutset property.*

Note that the top-bottom cutset $S(C)$ must contain a minimal top-bottom cutset. The following intuitively obvious lemma shows that in order to count the minimal top-bottom cutsets of $\mathcal{G}_{tri}(\ell,\ell)$ it is enough to look at all minimal right-left paths in $\mathcal{G}_{tri}(\ell,\ell)$. Its formal proof can be found in Appendix D.

Lemma 4.6 (Minimal Cutsets are Minimal Paths) *A set of nodes S in $\mathcal{G}_{tri}(\ell,\ell)$ is a minimal top-bottom cutset (resp. right-left cutset) if and only if it is a minimal right-left path (resp. top-bottom path).*

By Lemma 4.6, we conclude that if an I-avoiding top-bottom path doesn't exist for a colouring C then $S(C)$ contains a minimal right-left path $P_{c,x}$. Since $P_{c,x}$ is a subset of $S(C)$, its nodes only have colours in I. So, over the random choice of colouring C_{rand}, the probability that an I-avoiding top-bottom path doesn't exist is equal to the probability $p_x(I)$ that there exists a minimal right-left path $P_{c,x}$ whose node colours are all in t-collusion I.

Let $N_P(k,\ell)$ denote the total number of minimal right-left paths in $\mathcal{G}_{tri}(\ell,\ell)$ of length k. Since node colours are chosen independently and uniformly in $[n]$, each such path has probability $(t/n)^k$ to have all its node colours in I. It is clear that $\ell \leq k \leq \ell^2$. So, summing over all possible path lengths, we get the following upper bound: $p_x(I) \leq \sum_{k=\ell}^{\ell^2} N_P(k,\ell)(t/n)^k$. By symmetry, a similar argument gives the same upper bound on the probability $p_y(I)$ that a right-left I-avoiding path P_y does not exist. So we get the following upper bound on the probability $p(I)$ that either I-avoiding top-bottom path doesn't exist or an I-avoiding right-left path doesn't exist for each fixed t-subset I: $p(I) \leq 2\sum_{k=\ell}^{\ell^2} N_P(k,\ell)(t/n)^k$. Finally, taking a union bound over all $\binom{n}{t}$ possible t-colour subsets I, we get an upper bound on the probability p that the colouring C_{rand} is not Weakly t-Reliable of the form $p \leq 2\sum_{k=\ell}^{\ell^2} N_P(k,\ell)(t/n)^k\binom{n}{t}$. Using the bound on $N_P(k,\ell)$ from Lemma 4.5, we get

$$p \leq 2c(\mu)\ell^3(\mu t/n)^\ell \binom{n}{t}. \tag{6}$$

Since $n/t \geq R > \mu$, it is clear that this upper bound on p is less than 1 for sufficiently large ℓ. In fact, it suffices to take $\ell = O(\log(\binom{n}{t})/\log(n/(\mu t))) = O(n)$, as claimed. Now suppose we fix $\delta > 0$ and we want to find a lower bound on ℓ such that the error probability $p \leq \delta$. From (6) and using $n/t \geq R$ we see that $p \leq \delta$ is satisfied as long as

$$\ell\log(R/\mu) - 3\log(\ell) \geq \log(2c(\mu)N\delta^{-1}), \tag{7}$$

where $N = \binom{n}{t}$. Take $\ell = b\log(N)/\log(R/\mu)$. Plugging this choice of ℓ into (7), and using the fact that $N \geq \binom{\lceil R\rceil}{1} \geq R$ for all $n \geq R$ (since $N = \binom{n}{n/R}$ increases monotonically with n), we conclude

that (7) is satisfied if the constant b is sufficiently large such that (5) holds. This completes the proof. □

Combining Theorem 4.1 (applied with $n' = R \cdot t \leq n$ colours from $[n]$ for constant $R > \mu$) with Lemmas 4.1, 4.2, 4.4 and 4.5, we have

Corollary 4.2 *For any constant $R > 2.948$, if $t \leq n/R$, there exists a black box t-private protocol for f_G with communication complexity $O(nt^2)$ group elements. Moreover, for any $\delta > 0$, we can construct a probabilistic algorithm, with run-time polynomial in n and $\log(\delta^{-1})$, which outputs a protocol Π for f_G such that the communication complexity of Π is $O(nt^2 \log^2(\delta^{-1}))$ group elements and the probability that Π is not t-private is at most δ.*

Remark. Our computational experiments indicate that $t > n/2.948$ can be achieved with moderate values of ℓ – for example, for $n = 24$, $t = 11$ (i.e. $t \approx n/2.182$), we found a t-Reliable n-Colouring of $\mathcal{G}_{tri}(\ell, \ell)$ with $\ell = 350$, which is much smaller than $\binom{n}{t} \approx 2.5 \cdot 10^6$.

4.5 Generalisations and Other Results

General functions over G. Some applications may require n-party computation of more general functions over G (using only the group operation) instead of f_G. The most general such function is of the form $f'_G(x_1, \ldots, x_m) = x_1 \ldots, x_m$, where $m \geq n$ and each of the n parties holds one or more x_i's. Our reduction from Section 4.2 (and hence all our protocols) trivially extends to this most general case in the natural way.

General adversary structures. One may also consider more general adversary structures in place of the t-threshold structure. With the exception of our second construction in Section 4.4, all other results in the paper trivially generalise to the case of a Q^2 adversary structure $\mathcal{A}$, in which no pairwise union of collusions in $\mathcal{A}$ covers all n parties [14]. In particular, the generalisation of the first construction in Section 4.4 has communication complexity $O(n|\mathcal{A}|^2)$ group elements.

More efficient protocols for small t. For the cases $t \in \{1, 2\}$, we have managed to design explicit t-private black-box protocols for f_G with linear communication complexity ($O(n)$ group elements) and optimal collusion resistance. These protocols and their analysis can be found in Appendices E and F. We have also implemented a computer program for finding t-Reliable n-Colourings of a given graph, with which one can easily construct efficient protocols for small values of n, t (avoiding the error probability δ of Theorem 4.1).

General Boolean Functions. Here, we briefly review a method due to Barrington [3], which shows how to efficiently transform an arbitrary Boolean circuit C (consisting of AND and NOT gates) into a circuit C' over the non-Abelian group S_5 (consisting of S_5 multiplication gates), which computes the same Boolean function. We then explain a variant of our reduction from Section 4.2 showing how to reduce the secure computation of C' to the same shared 2-product subprotocol. This reduction demonstrates how our techniques can be applied to general secure multiparty computation.

In the following, for a group G, we define an n-input 1-output G-circuit C as a circuit (directed acyclic graph) with n input nodes, one output node, and two types of gates (corresponding to all other circuit nodes):

1. Mult: Given two inputs x and y in G, the gate output is $x \cdot y \in G$.
2. $\mathsf{CMult}_{\alpha,\beta}$: Given one input $x \in G$, the gate output is $\alpha \cdot x \cdot \beta \in G$ (note that the constants $\alpha, \beta \in G$ are built into the gate).

We denote by $f_C : G^n \to G$ the function computed by the G-circuit C. Let 1_G denote the identity element of G. For some fixed $\sigma \in G \setminus \{1_G\}$, let $\phi_\sigma : \{0, 1\} \to G$ denote the encoding function mapping 0 to 1_G and 1 to σ. We say that a G-circuit C *computes* a Boolean function g if there exists $\sigma \in G$ such that $g(x_1, \ldots, x_n) = \phi_\sigma^{-1}(f_C(\phi_\sigma(x_1), \ldots, \phi_\sigma(x_n)))$ for all $(x_1, \ldots, x_n) \in \{0, 1\}^n$.

Since Barrington's result is presented in a different context than in [3], we provide a proof adapted from [3] for completeness.

Theorem 4.2 (Adapted from [3]) *Let C be a Boolean circuit consisting of N_A 2-input AND gates, N_N NOT gates, and depth d. Then there exists an S_5-circuit C' which computes the Boolean function computed by C. The circuit C' contains $N'_M = 3N_A$ Mult gates and $N'_{CM} = 4N_A + N_N$ CMult gates, and has depth $d' \leq 4d$.*

Proof. It suffices to show an S_5-circuit for computing the AND function (using 3 Mult gates and 4 CMult gates) and another for computing the NOT function (using 1 CMult gate).

We recall that two elements $x, y \in S_5$ are called *conjugates* if there exists $h \in S_5$ such that $x = h \cdot y \cdot h^{-1}$. It is easy to check that conjugacy is an equivalence relation, and hence partitions S_5 into conjugacy equivalence classes. Barrington's method is based on the conjugacy class J of all 5-cycles of S_5. In particular, J contains two distinct elements $\sigma_1 = (12345)$ and $\sigma_2 = (13542)$ whose commutator $c = [\sigma_1, \sigma_2] = \sigma_1\sigma_2\sigma_1^{-1}\sigma_2^{-1} = (13254)$ is also in J. Furthermore, it is clear that σ_1^{-1} is also in J.

For $x \in \{0, 1\}$, and $\sigma \in J$, let $x_\sigma = \phi_\sigma(x)$ denote the encoding relative to σ. First, we observe that for $\sigma, \sigma' \in J$, we can convert an encoding of x relative to σ to an encoding of x relative to σ' using one CMult gate, namely $x_{\sigma'} = h_{\sigma,\sigma'} x_\sigma h_{\sigma,\sigma'}^{-1}$, where $h_{\sigma,\sigma'}$ satisfies $\sigma' = h_{\sigma,\sigma'}\sigma h_{\sigma,\sigma'}^{-1}$ and exists by conjugation of σ, σ'.

The AND function $z = AND(x, y)$ can be computed by the following S_5 circuit, relative to the encoding ϕ_{σ_1}. Given inputs $x_{\sigma_1}, y_{\sigma_1} \in S_5$:

- Compute by encoding conversion (using 3 CMult gates) $x_{\sigma_1^{-1}}$, y_{σ_2}, $y_{\sigma_2^{-1}}$.
- Compute (using 3 Mult gates) $z_c = x_{\sigma_1} y_{\sigma_2} x_{\sigma_1^{-1}} y_{\sigma_2^{-1}}$ (note that $z_c = [x_{\sigma_1}, y_{\sigma_2}]$ is an encoding of $z = AND(x, y)$ relative to $c = [\sigma_1, \sigma_2]$).
- Compute by encoding conversion (using a CMult gate) z_{σ_1}.

The NOT function can be computed using one CMult gate because $NOT(x)_{\sigma_1^{-1}} = x_{\sigma_1} \cdot \sigma_1^{-1}$. The composition of multiplication by σ_1^{-1} and the encoding conversion from σ_1^{-1} to σ_1 can be combined into one CMult gate. □

Next, we construct a protocol $\prod(C', \prod_S^{(x)}, \prod_S^{(y)})$ for private computation of the S_5-circuit C' with n input nodes corresponding to the n protocol inputs, using a pair of compatible shared 2-product subprotocols $\prod_S^{(x)}$ and $\prod_S^{(y)}$ (see Definition 4.2) by a variant of the reduction from Section 4.2. We assume that $\prod_S^{(x)}$ (resp. $\prod_S^{(y)}$) satisfy x-preserving (resp. y-preserving) strong t-privacy, with sharing parameter ℓ and share ownership functions $\mathcal{O}_x, \mathcal{O}_y, \mathcal{O}_z$.

To simplify the protocol description, we assume the default sharing of node values is an $\mathcal{O}_x$-sharing, which is converted to an $\mathcal{O}_y$-sharing when required using the following conversion subprotocol:

- Subprotocol Convert ($\mathcal{O}_x$ to $\mathcal{O}_y$ Sharing Conversion): Given an $\mathcal{O}_x$ sharing $(s_x(1), \ldots, s_x(\ell))$ of x, where party $P_{\mathcal{O}_x(j)}$ holds $s_x(j)$ for $j = 1, \ldots, \ell$. P_1 (or any other arbitrary party) computes a random sharing $s_y(1) \cdots s_y(\ell) = 1_{S_5}$ of the identity element, and sends $s_y(j)$ to party $P_{\mathcal{O}_y(j)}$ for $j = 1, \ldots, \ell$. Then shared 2-product subprotocol $\prod_S^{(y)}$ is run on the shared x value (as the left input to $\prod_S^{(y)}$), and the shared 1_{S_5} value (as the right input to $\prod_S^{(y)}$), resulting in an $\mathcal{O}_y$ sharing of x.

The protocol $\prod$ begins with each party P_j computing an ℓ-of-ℓ sharing of its input x_j (x_j is assigned to be the value of the jth input node of C'), and distributing these shares to the n parties according to the share ownership function $\mathcal{O}_x$. Then, for each internal node N of C':

- If node N is a Mult gate with incoming $\mathcal{O}_x$-shared values $x = s_x(1) \cdots s_x(\ell)$ and $y = s_y(1) \cdots s_y(\ell)$, run subprotocol Convert to convert the $\mathcal{O}_x$-sharing $s_y(1) \cdots s_y(\ell)$ of y to an $\mathcal{O}_y$-sharing $s'_y(1) \cdots s'_y(\ell)$ of y. Then, run 2-product subprotocol $\prod_S^{(x)}$ on the sharings $(s_x(1), \ldots, s_x(\ell))$ and $(s'_y(1), \ldots, s'_y(\ell))$, resulting in an $\mathcal{O}_x$-sharing of node N's output value $x \cdot y$.

- If node N is a $\mathsf{CMult}_{\alpha,\beta}$ gate with incoming $\mathcal{O}_x$-shared value $x = s_x(1) \cdots s_x(\ell)$, party $\mathcal{O}_x(1)$ replaces its input share $s_x(1)$ with $\alpha \cdot s_x(1)$, and party $\mathcal{O}_x(\ell)$ replaces its input share $s_x(\ell)$ with $s_x(\ell) \cdot \beta$.

Eventually, this recursive process gives an ℓ-of-ℓ sharing of the output node value of C', which is broadcast to all parties.
The following lemma establishes the t-privacy of protocol $\prod(C', \prod_S^{(x)}, \prod_S^{(y)})$, assuming the correctness and strong t-privacy of subprotocols $\prod_S^{(x)}$,$\prod_S^{(y)}$. The proof is similar to that of Lemma 4.1.

Lemma 4.7 *For any S_5-circuit C' with n input nodes, if the n-party compatible shared 2-product subprotocols $\prod_S^{(x)}$ (resp. $\prod_S^{(y)}$) satisfy correctness and x-preserving (resp. y-preserving) strong t-privacy (see Definition 4.1 and Definition 4.2), then the protocol $\prod(C', \prod_S^{(x)}, \prod_S^{(y)})$ is an n-party t-private protocol for computing the function $f_{C'}(x_1, \ldots, x_n)$.*

Combining Theorem 4.2, Lemma 4.7 and Theorem 4.1, we obtain the following.

Corollary 4.3 *Let C be an AND/NOT Boolean circuit with N_A AND gates. For any $\delta > 0$, if $t < n/2.948$, we can construct a probabilistic algorithm, with run-time polynomial in n and $\log(\delta^{-1})$, which outputs a protocol $\prod$ for C such that the communication complexity of $\prod$ is $O((n + \log \delta^{-1})^2 \cdot N_A)$ bits.*

5 Conclusions

We showed how to design black-box t-private protocols for computing the n-product function over any finite group by reducing the problem to a combinatorial graph colouring problem, using tools from communication security [10]. Our work raises some interesting combinatorial questions. For example, for our PDAG $\mathcal{G}_{tri}(\ell, \ell)$, what is the shape of the 'tradeoff' curve $R_{max}(\ell)$ relating the maximal achievable (using a suitable colouring) secure collusion resistance $R_{max} = t/n$ to the graph size ℓ? (we showed that $R_{max}(\ell) \geq 1/2.948$ for $\ell = O(t)$ and $R_{max}(\ell) = 1/2$ for $\ell \geq \binom{2t+1}{t}$). More generally, what is the largest collusion resistance achievable with an admissible PDAG of size polynomial in n, and what kind of PDAG achieves this optimum? There are also interesting cryptographic questions. First, can our black-box protocols be efficiently strengthened to yield black-box protocols secure against *active* adversaries? Second, can the communication complexity $O(nt^2)$ of our t-private protocols be reduced further? Third, does there exist an efficient (run-time polynomial in n) *deterministic* algorithm to generate a Weakly t-Reliable n-Colouring of $\mathcal{G}_{tri}(\ell, \ell)$ (or some other admissible PDAG) given n, t as input?

Acknowledgements. This research was supported by ARC research grants DP0451484, DP0558773, DP0663452 and DP0665035. Ron Steinfeld's research was supported in part by a Macquarie University Research Fellowship (MURF). Huaxiong Wang's research was supported in part by the Singapore Ministry of Education grant (T206B2204). Yvo Desmedt is grateful for the research visits to Macquarie University. The authors also thank Chris Charnes and Scott Contini for helpful discussions about this work, and Yuval Ishai for pointing out the applicability of our results to general multiparty computation via the work of Barrington [3]. We thank Christophe Tartary for his assistance with Sec. 4.5.

References

[1] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley-Interscience, 2000.

[2] J. Bar-Ilan and D. Beaver. Non-Cryptographic Fault-Tolerant Computing in a Constant Number of Rounds of Interaction. In *Symposium on Principles Of Distributed Computing (PODC)*, pages 201–209, New York, 1989. ACM.

[3] D.A. Barrington. Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1. In *Proceedings of the eighteenth annual ACM Symp. Theory of Computing, STOC*, pages 1–5, 1986.

[4] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *Proc. 20-th STOC*, pages 1–10. ACM, 1988.

[5] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM Symp. Theory of Computing, STOC*, pages 11–19, May 2–4, 1988.

[6] H. Chen and R. Cramer. Algebraic Geometric Secret Sharing Schemes and Secure Multi-Party Computations over Small Fields. In *CRYPTO 2006*, volume 4117 of *LNCS*, pages 521–536, Berlin, 2006. Springer-Verlag.

[7] H. Chen, R. Cramer, S. Goldwasser, R. de Haan, and V. Vaikuntanathan. Secure Computation from Random Error Correcting Codes. In *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 291–310, Berlin, 2007. Springer-Verlag.

[8] R. Cramer, S. Fehr, Y. Ishai, and E. Kushilevitz. Efficient Multi-Party Computation Over Rings. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 596–613, Berlin, 2003. Springer-Verlag.

[9] Y. Desmedt, J. Pieprzyk, R. Steinfeld, and H. Wang. On Secure Multi-Party Computation in Black-Box Groups. Full version of this paper, 2007. Available at `http://www.comp.mq.edu.au/~rons/`.

[10] Y. Desmedt, Y. Wang, and M. Burmester. A Complete Characterization of Tolerable Adversary Structures for Secure Point-to-Point Transmissions. In *ISAAC 2005*, volume 3827 of *LNCS*, pages 277–287, Berlin, 2005. Springer-Verlag.

[11] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Trans. on Information Theory*, 22:644–654, 1976.

[12] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Tran. Info. Theory*, IT-31(4):469–472, 1985.

[13] O. Goldreich. *Foundations of Cryptography, Volume II.* Cambridge University Press, Cambridge, 2004.

[14] M. Hirt and U. Maurer. Complete Characterization of Adversaries Tolerable in Secure Multi-Party Computation (Extended Abstract). In *Symposium on Principles Of Distributed Computing (PODC)*, pages 25–34, New York, 1997. ACM.

[15] E. Kushilevitz. Privacy and Communication Complexity. *SIAM J. on Discrete Mathematics*, 5(2):273–284, 1992.

[16] S. Magliveras, D. Stinson, and T. van Trung. New approaches to Designing Public Key Cryptosystems using One-Way Functions and Trapdoors in Finite Groups. *Journal of Cryptology*, 15:285–297, 2002.

[17] J. Noonan. New Upper Bounds for the Connective Constants of Self-Avoiding Walks. *Journal of Statistical Physics*, 91(5/6):871–888, 1998.

[18] S. Paeng, K. Ha, J. Kim, S. Chee, and C. Park. New Public Key Cryptosystem Using Finite Non Abelian Groups. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 470–485, Berlin, 2001. Springer-Verlag.

[19] A. Pönitz and P. Tittmann. Improved Upper Bounds for Self-Avoiding Walks in $\mathbb{Z}^d$. *The Electronic Journal of Combinatorics*, 7, 2000.

[20] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–128, 1978.

[21] A. Shamir. How To Share a Secret. *Communications of the ACM*, 22:612–613, 1979.

[22] P. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comp.*, 26(5):1484–1509, 1997.

A Precise Specification of Protocol $\Pi(T, \Pi_S)$

For the specification of Π below, we use the following notation. Let σ_T denote the number of nodes in the binary tree T. Assume that the nodes of T are indexed by integers $1, 2, \ldots, \sigma_T$, such that (1) The n leaf nodes of T are indexed by the integers $\{1, \ldots, n\}$ from left to right (such that leftmost leaf of T is node 1 and for $i = 1, \ldots, n$, the ith leaf node from the left is node i), and (2) Internal nodes of T have indices greater than the indices of both their children nodes. For each node i of T (except the root node), we call i an L node (resp. R node) if i is a left child (resp. right child) of the parent node of i. For a parent node i, we denote by $L(i)$ (resp. $R(i)$) the left (resp. right) child node of i.

n-Product Protocol $\Pi(T, \Pi_S^{(x)}, \Pi_S^{(y)})$

Input: For $i = 1, \ldots, n$, party P_i holds input $x_i \in G$.

1. For $i = 1, \ldots, n$, party P_i generates a uniformly random ℓ-of-ℓ sharing $\boldsymbol{s}_{x_i} = (s_{x_i}(1), \ldots, s_{x_i}(\ell))$ of $x_i = s_{x_i}(1) \cdots s_{x_i}(\ell)$. If leaf node i of T is an L node (resp. R node), then for $j = 1, \ldots, \ell$, P_i sends jth share $s_{x_i}(j)$ to party $P_{\mathcal{O}_x(j)}$ (resp. $P_{\mathcal{O}_y(j)}$). Label leaf node i of T with $\boldsymbol{s}_{x_i}$.

2. For each internal node $i = n+1, \ldots, \sigma_T$ of T, parties $P_1, \ldots, P_n$ run subprotocol $\Pi_S^{(c)}$ (where $c = x$ if node i is an L node or the root node, and $c = y$ if i is an R node) on subprotocol inputs $\boldsymbol{s}_x^{(i)}$ and $\boldsymbol{s}_y^{(i)}$ being the labels of the left (resp. right) children nodes $L(i)$ (resp. $R(i)$) of node i (note that at this stage, for $j = 1, \ldots, \ell$, party $P_{\mathcal{O}_x(j)}$ holds jth share $s_x^{(i)}(j)$ of $\boldsymbol{s}_x^{(i)}$ and party $P_{\mathcal{O}_y(j)}$ holds jth share $s_y^{(i)}(j)$ of $\boldsymbol{s}_y^{(i)}$). Label node i with the output shares $\boldsymbol{s}_z^{(i)}$ of the subprotocol $\Pi_S^{(c)}$. At the end of the subprotocol run, for each $j = 1, \ldots, \ell$, party $P_{\mathcal{O}_z(j)}$ holds an output share $s_z^{(i)}(j)$. If node i is an L node (resp. R node), then for $j = 1, \ldots, \ell$, $P_{\mathcal{O}_z(j)}$ sends jth output share $s_z(j)$ to party $P_{\mathcal{O}_x(j)}$ (resp. $P_{\mathcal{O}_y(j)}$) to prepare for next subprotocol run. Otherwise, if i is the root node, go to the next step.

3. For each $j = 1, \ldots, \ell$, party $P_{\mathcal{O}_z(j)}$ broadcasts to all parties the share $s_z^*(j)$, where $\boldsymbol{s}_z^* = (s_z^*(1), \ldots, s_z^*(\ell))$ is the label of the root node of T.

4. All parties compute protocol output $y = s_z^*(1) \cdots s_z^*(\ell)$.

B Proof of Lemma 4.1

The correctness of $\Pi_S^{(x)}$ and $\Pi_S^{(y)}$ implies that the label of each internal node α in T is a ℓ-of-ℓ sharing of the product of the values whose sharings are the labels of the children nodes of α. It follows easily by induction that the label of the root node of T is a ℓ-of-ℓ sharing of $y = x_1 \cdots x_n$, establishing the correctness of $\Pi(T, \Pi_S^{(x)}, \Pi_S^{(y)})$.

To establish the t-privacy, we construct a simulator algorithm S for the view of a t-collusion $I \subset [n]$. Given I, $\{x_i\}_{i \in I}$ and protocol output $y = x_1 \cdots x_n$, the algorithm S runs as follows.

1. For each leaf node $i = 1, \ldots, n$ of T, S chooses $\ell - 1$ independent uniformly random group elements to simulate the $\ell - 1$ shares $\{s_{x_i}(j)\}_{j \in [\ell] \setminus \{j^*\}}$ of x_i sent out by party P_i to parties $P_{\mathcal{O}_c(j)}$ (where $c = x$ if i is an L node and $c = y$ if i is an R node) for $j \in [\ell] \setminus \{j_c^*\}$, where $j_x^*, j_y^* \in [\ell]$ (which depend on I) are the share indices which are *not* simulated/input by/to simulator S_{Π_S} of subprotocols $\Pi_S^{(x)}$ and $\Pi_S^{(y)}$ (see Def. 4.1). S labels leaf node i with $\{s_{x_i}(j)\}_{j \in [\ell] \setminus \{j_c^*\}}$.

2. For each internal node $i = n+1, \ldots, \sigma_T$ of T, S runs simulator S_{Π_S} of subprotocol $\Pi_S^{(c)}$ (where $c = x$ if i is an L node and $c = y$ if i is an R node) on input $(I, \{s_x(j)\}_{j \in [\ell] \setminus \{j_x^*\}}, \{s_y(j)\}_{j \in [\ell] \setminus \{j_y^*\}})$, where $\{s_x(j)\}_{j \in [\ell] \setminus \{j_x^*\}}, \{s_y(j)\}_{j \in [\ell] \setminus \{j_y^*\}}$ are the labels of the left and right children nodes of node i, respectively. Simulator $\mathsf{S}_{\Pi_S^{(c)}}$ outputs a simulation of the internal view $\langle \mathsf{VIEW}_I^{\Pi_S(i)} \rangle$ of I in $\Pi_S^{(c)}$ at node i, along with simulated output shares (except the $j*_c$th) $\{s_z^{(i)}(j)\}_{j \in [\ell] \setminus \{j_c^*\}}$ of $\Pi_S^{(c)}$. S labels node i of T with $\{s_z^{(i)}(j)\}_{j \in [\ell] \setminus \{j_c^*\}}$.

3. After labelling all nodes of T, S has a complete simulation of the view of I in $\Pi(T, \Pi_S^{(c)})$, except for the j^*th share $s_z^*(j^*)$ of root node label, broadcast by party $P_{\mathcal{O}_z(j^*)}$ in the last round. However, we know that the root node shares satisfy $s_z^*(1) \cdots s_z^*(\ell) = x_1 \cdots x_n \stackrel{\text{def}}{=} y$. Accordingly, from the known value of y and the simulated shares $\{s_z(j)\}_{j\in[\ell]\setminus\{j^*\}}$, $s_z^*(j^*)$ is uniquely determined and simulated by S by computing it as follows:

$$s_z^*(j^*) = (s_z^*(1) \cdots s_z^*(j^*-1))^{-1} \cdot y \cdot (s_z^*(j^*+1) \cdots s_z^*(\ell))^{-1}.$$

Note that for each $i \in [n]$, the simulation of the $\ell - 1$ shares sent in the first round by P_i is perfect due to the perfect ℓ-of-ℓ secret sharing used by P_i to share x_i. Furthermore, the unsimulated share $s_{x_i}(j_c^*)$ of x_i is not in the view of I since $\mathcal{O}_c(j_c^*) \notin I$.

For each internal node $i > n$ of T, the perfect simulation of $\langle \mathsf{VIEW}_I^{\Pi_S(i)}, \{s_z^{(i)}(j)\}_{j\in[\ell]\setminus\{j^*\}}\rangle$ given the view so far follows inductively from the strong t-privacy of $\Pi_S^{(c)}$, and the fact that the internal view in $\Pi_S^{(c)}$ at node i depends only on the inputs to $\Pi_S^{(c)}$ at node i, that is:

$$\begin{aligned}
&\Pr[\langle \mathsf{VIEW}_I^{\Pi_S(i)}, \{s_z^{(i)}(j)\}_{j\in[\ell]\setminus\{j_c^*\}}\rangle | \{\langle \mathsf{VIEW}_I^{\Pi_S(k)}, \{s_z^{(k)}(j)\}_{j\in[\ell]\setminus\{j^*\}}\rangle\}_{k=1,\ldots,i-1}] \\
&= \Pr[\langle \mathsf{VIEW}_I^{\Pi_S(i)}, \{s_z^{(i)}(j)\}_{j\in[\ell]\setminus\{j_c^*\}}\rangle | \{s_x^{(i)}(j)\}_{j\in[\ell]\setminus\{j_x^*\}}, \{s_y^{(i)}(j)\}_{j\in[\ell]\setminus\{j_y^*\}}] \\
&= \Pr[\mathsf{S}_{\Pi_S^{(c)}}(\{s_x^{(i)}(j)\}_{j\in[\ell]\setminus\{j_x^*\}}, \{s_y^{(i)}(j)\}_{j\in[\ell]\setminus\{j_y^*\}}) \\
&\qquad\qquad | \{s_x^{(i)}(j)\}_{j\in[\ell]\setminus\{j_x^*\}}, \{s_y^{(i)}(j)\}_{j\in[\ell]\setminus\{j_y^*\}}]. \qquad (8)
\end{aligned}$$

It follows that the full view of I in $\Pi(T, \Pi_S^{(x)}, \Pi_S^{(y)})$ is perfectly simulated by S, as claimed. The complexity claims are easy to verify. This proves the general result assuming *symmetric* strong t-Privacy of Π_S.

□

C Proof of Lemma 4.2

The correctness of $\Pi_S(\mathcal{G}, C)$ is immediate from the fact that, due to the ordering condition (1) in Definition 4.3, the product of node labels at each row of $\mathcal{G}$ is preserved to be equal to $x \cdot y$.

To establish the strong t-privacy of $\Pi_S(\mathcal{G}, C)$, let $I \subseteq [n]$ denote an arbitrary t-colour collusion. Since C is t-reliable there exists an I-avoiding path $PATH_x$ in $\mathcal{G}$ from j_x^*th x-input node to the j_z^*th output node, and an I-avoiding path $PATH_y$ from the j_y^*th y-input to the j_z^*th output node. On input $(I, \{\boldsymbol{s}_x(j)\}_{j\in[n]\setminus\{j_x^*\}}, \{\boldsymbol{s}_y(j)\}_{j\in[\ell]\setminus\{j_y^*\}})$, the simulator S_{Π_S} runs as follows:

- For each $i = 1, \ldots, m$ and $j = 1, \ldots, m$:
  - If node (i, j) is not on $PATH_x$ or $PATH_y$, follow the protocol Π_S to label node (i, j) and all its outgoing edges (using the product of incoming edge values if $i > 1$ or with appropriate simulator input if $i = 1$).
  - If node (i, j) is on $PATH_x$ or $PATH_y$, label all outgoing edges of node (i, j) which go to nodes not on $PATH_x$ or $PATH_y$ by independent random elements of G.

Since $PATH_x$ and $PATH_y$ are I-avoiding, the edge values simulated by S_{Π_S} contain the view of I, and S_{Π_S} also simulates the values of all output nodes except the j_z^*th output (which is the only output node on $PATH_x$ or $PATH_y$).

Let V denote a fixed value for the labels of edges NOT on $PATH_x$ or $PATH_y$. We show that, in the real subprotocol Π_S, for each fixed choice of the 'missing inputs' $s_x(j_x^*)$ and $s_y(j_y^*)$ there exists a unique value for the random group elements $\{r_i\}$ chosen at nodes on $PATH_x$ and $PATH_y$, which is consistent with the view V. Thus the simulation is perfect, as required.

To each outgoing edge of a node on $PATH_x$ or $PATH_y$ (we call such a node a 'path node' from now on) in the real subprotocol Π_S we associate an *edge equation* relating the label of the edge to the random elements $\{r_i\}$ chosen by path nodes, the fixed view V, and the values $s_x(j_x^*)$ and $s_y(j_y^*)$. For an outgoing edge of a node on the path, we say that its edge equation is a *view edge equation* if the edge is not on the path. Our problem is therefore to show that the collection of view edge equations always has a unique solution for the $\{r_i\}$ (for any fixed values of V, $s_x(j_x^*)$ and $s_y(j_y^*)$). To do this, we show that one can order the view edge equations $\{E_i\}$ and the random elements $\{r_i\}$ chosen by path nodes, so that for all i, the following 'good ordering' condition is satisfied:

- **Good Ordering Condition:** The ith view edge equation E_i is of the form

$$\alpha_1 r_i \alpha_2 = \alpha_3, \tag{9}$$

  where r_i is *new* (i.e. r_i does not appear in any of the first $i-1$ view edge equations $E_1, \ldots, E_{i-1}$), while $\alpha_1, \alpha_2, \alpha_3$ are group elements determined by 'old' r_i's (i.e. $r_1, \ldots, r_{i-1}$) and the fixed values V, $s_x(j_x^*)$ and $s_y(j_y^*)$.

If such a good ordering exists, it is clear that a unique solution for the $\{r_i\}$ exists (where for all i, the ith view equation E_i uniquely determines $r_i = \alpha_1^{-1}\alpha_3\alpha_2^{-1}$ in terms of already determined previous r_j's and fixed values).

We now construct a good ordering of the view edge equations $\{E_i\}$ and path random values $\{r_i\}$.

First observe that since $PATH_x$ and $PATH_y$ both exit at output node j_z^*, the two paths must meet at some node, and then continue along the same path to output j_z^*. Except for the unique *meeting* node where $PATH_x$ and $PATH_y$ meet, we will regard the common nodes as belonging to $PATH_x$. Apart from the meeting node, we can classify the nodes on $PATH_x$ and $PATH_y$ into three classes: *minimum* path nodes (which have two incoming edges on the path and no outgoing edges on the path), *maximum* path nodes (which have no incoming edges on the path and two outgoing edges on the path) and *middle* path nodes (which have one incoming and one outgoing edge on the path, or are input/output nodes). Refer to Figure 6 for an example.

Let us first consider the nodes on $PATH_y$. By Proposition 4.1, we may assume (without changing the protocol distribution) that in the real subprotocol Π_S, when a middle $PATH_y$ node shares out its node label v among its q outgoing edges, it sends new random elements (labels) r_i on each of the $q-1$ outgoing edges NOT on $PATH_y$. We take the view edge equations corresponding to those nodes to be the first edge equations in our good ordering. It is clear that the good ordering condition (9) is trivially satisfied, since these equations have the form $r_i = \alpha$, where r_i is new and α is fixed by V.

Suppose there are k minimum/maximum nodes on the $PATH_y$ (since each minimum node is always followed by a maximum node, the number of maxima and minima is always equal). Note that if the meeting node at the intersection between $PATH_x$ and $PATH_y$ has an outgoing (downward) edge along $PATH_y$, we regard it here as the last maximum node of $PATH_y$. For each $j = 1, \ldots, k$, applying Proposition 4.1 again to the jth maximum path node having $q \geq 2$ outgoing edges, we can assume that the $q-2$ outgoing edges NOT on the path are labelled with new random elements r_i. We take the corresponding view edge equations as the next ones in our ordering, again trivially satisfying (9). Of the two outgoing edges on the path from the jth maximum node, one of them goes to an *earlier* node on the path (towards the jth 'minimum' path node), and we may assume that this edge is labelled with a new $(q-1)$th random element r_j^* (note that r_j^* does NOT appear in any view edge equation so far and is left undetermined for now). The remaining outgoing edge on the path from the jth maximum therefore has a label of the form $\alpha_1(r_j^*)^{-1}\alpha_2$, where α_1 and α_2 are fixed by V and the old r_i's. Therefore, it is easy to see that for each $j = 1, \ldots, k$ the label $v_{j,j}$ on any path edge between the jth minimum and the jth maximum is of the form

$$v_{j,j} = \alpha_1 r_j^* \alpha_2, \tag{10}$$

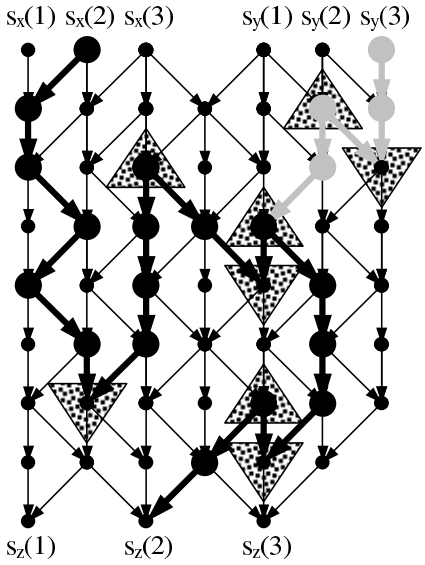

Figure 6: Example of an admissible PDAG $\mathcal{G}$ with sharing parameter $\ell = 3$ (node colours are not indicated). For a given collusion I, an example I-avoiding path $PATH_x$ is shown in heavy black, and an example I-avoiding path $PATH_y$ (until the meeting with $PATH_x$) is shown in heavy gray. Minimum path nodes are marked by a dotted triangle with a horizontal edge above the node, whereas maximum path nodes are marked by a dotted triangle with a horizontal edge below the node. In this example, we have $j_x^* = 2$ and $j_y^* = 3$, $PATH_x$ has 3 maximum and 3 minimum nodes, and $PATH_y$ has 1 maximum and 1 minimum node.

where α_1 and α_2 are fixed by V. Similarly, for $j = 1, \ldots, k-1$, the label $v_{j,j+1}$ on any path edge between the jth maximum and and the $(j+1)$th minimum is of the form

$$v_{j,j+1} = \beta_1 (r_j^*)^{-1} \beta_2, \tag{11}$$

where β_1 and β_2 are fixed by V.

Suppose we have visited the first $j-1$ minimum nodes and consider the jth minimum node. Applying Proposition 4.1 to the jth 'minimum' path node for $j = 1, \ldots, k$, we see that out of the q outgoing edges not on the path of the jth minimum node, we can assume that $q-1$ of those outgoing edge labels are new random elements r_i, giving $q-1$ new edge view equations trivially satisfying (9), and it remains to consider the the view edge equation corresponding to the qth outgoing edge label, which is determined from all other incoming and outgoing edges of the jth minimum node. This equation has the form

$$\gamma_1 v_{j-1,j}^{\pm 1} \gamma_2 v_{j,j}^{\pm 1} \gamma_3 = \gamma_4, \tag{12}$$

where γ_i's are fixed by V, $v_{j-1,j}$ is the label on the path edge entering the jth minimum node, and $v_{j,j}$ is the value on the path edge exiting the jth minimum node.

For $j = 1$, the label $v_{0,1}$ on the incoming path edge to the first minimum node from an *earlier* node on the path (i.e. towards the input node) has the form $v_{0,1} = \beta_1 s_y(j_y^*)\beta_2$, with β_1, β_2 fixed by V. Using (10), the label $v_{1,1}$ on the incoming path edge to the first minimum node from the next node on the path (i.e. towards the first maximum node) has the form $v_{1,1} = \alpha_1 r_1^* \alpha_2$, with α_1, α_2 fixed by V. Plugging these expressions for $v_{0,1}$ and $v_{1,1}$ into (12), we see that since r_1^* is 'new' (recall that r_1^* has only appeared in equations of edges *on the path* so far, and not in any *view* edge equation), the condition (9) is satisfied for $j = 1$.

For $j > 1$, using (11), we have $v_{j-1,j} = \beta_1 (r_{j-1}^*)^{-1} \beta_2$ with β_1, β_2 fixed by V, while using (10), we have $v_{j,j} = \alpha_1 r_j^* \alpha_j$, with α_1, α_2 fixed by V. Plugging these expressions for $v_{j-1,j}$ and $v_{j,j}$ into (12), and noting that r_{j-1}^* is 'old' (has appeared in a view edge equation for an outgoing edge of the $(j-1)$th minimum node) while r_j^* is 'new', we see that condition (9) is also satisfied for $1 < j \leq k$.

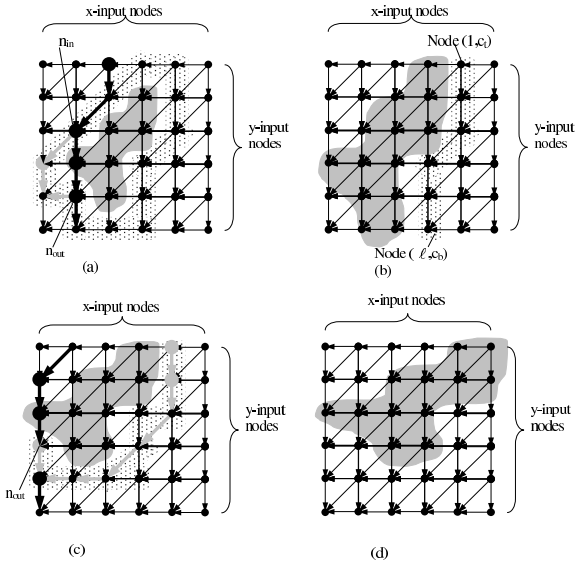

Figure 7: The four subcases of Lemma 4.6: (a) Subcase 1, (b) Subcase 2, (c) Subcase 3, (d) Subcase 4. In each subcase, the gray shaded region shows the connected set S_1, the contracted path P' is indicated by the dotted region, the edges of path P_x are shown in heavy black, and the edges on the portion of path P'_x which goes along P' are shown in heavy gray.

This completes the description of the good ordering of all view edge equations of nodes on $PATH_y$. We now apply the same argument as above to add the view edge equations of nodes on $PATH_x$ in a good ordering. The only differences in this case are that (1) $s_x(j)$ takes the place of $s_y(j_y^*)$, and (2) the effect of the 'meeting' node of $PATH_x$ and $PATH_y$. The only effect of (2) on the above argument is that in the qth view edge equation (12) for the first minimum node on $PATH_x$ that follows the meeting node, one of the α_i's will depend also on the r_k^* random value which appeared in the $PATH_y$ view edge equations. Since the r_k^* is 'old', the good ordering condition (9) is still satisfied.

We conclude that all random values of path nodes are determined uniquely, for any fixed values of $s_x(j_x^*)$,$s_y(j_y^*)$ and view V. This completes the proof of the Lemma. □

D Proof of Lemma 4.6

The 'if' direction is immediate, since by planarity of $\mathcal{G}_{tri}(\ell,\ell)$, a right-left path must intersect any top-bottom path at some node of $\mathcal{G}_{tri}(\ell,\ell)$.

For the 'only if' direction, suppose, towards a contradiction, that a set of nodes S is NOT a minimal right-left path in $\mathcal{G}_{tri}(\ell,\ell)$ but is a minimal top-bottom cutset.

If S is a non-minimal right-left path, then it is possible to remove a node from S and still obtain a right-left path, which, as explained in the 'if direction' above is still a top-bottom cutset, contradicting the assumption that S is a minimal top-bottom cutset, as required.

The other case is that S is not a right-left path. In this case, if S is disconnected, it can be partitioned into $m \geq 1$ mutually disconnected but internally connected subsets $S_1,\ldots,S_m$. Consider the connected subset S_1. Let r_{min}, r_{max} denote the minimum (resp. maximum) row index over all nodes in S_1, and similarly let c_{min}, c_{max} denote the minimum (resp. maximum) column index over all nodes in S_1. There are several subcases:

Subcase 1: $2 \leq r_{min} \leq r_{max} \leq \ell - 1$ **and** $2 \leq c_{min} \leq c_{max} \leq \ell - 1$**.** In this case, the closed rectangular path P in $\mathcal{G}_{tri}(\ell,\ell)$ connecting the 4 corner nodes $(r_{min}-1, c_{min}-1)$,$(r_{min}-1, c_{max}+1)$,$(r_{max}+1, c_{max}+1)$,$(r_{max}+1, c_{min}-1)$ clearly has the following two properties:

(1) The interior region enclosed by closed path P contains S_1.

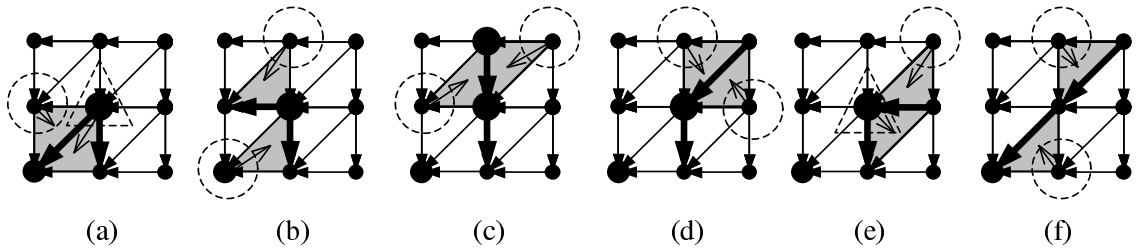

Figure 8: (a)-(f): The six types of contraction operations. In each case, the middle node is $P(j)$ and the edges connecting it to $P(j-1)$ and $P(j+1)$ are shown in heavy black. See text for further details.

(2) No node on P is in S_1.

Claim 1. The closed path P can be converted into a closed path P' that satisfies the above two properties of P as well as the following third property:

(3) Each node on P' is adjacent to a node in S_1 (we say that two nodes are *adjacent* if there is an edge between them).

Before we prove this claim, we note that property (3) of P' implies that no node on P' is in S_i for any $i \geq 2$, since otherwise S_1 would be connected to S_i, contrary to assumption. Together with property (2), we conclude that no node on P' is in S. On the other hand, let n denote any node of S_1. By minimality of S, we know that there exists a top-bottom path P_x which passes via node n, whereas all other nodes on P_x are not in S. Since by property (1), P' encloses S_1, it follows by planarity of $\mathcal{G}_{tri}(\ell, \ell)$ that P_x must enter the region enclosed by P' via some node n_{in} on P' and exit this region via some other node n_{out} on P'. But this means that we can modify top-bottom path P_x into another top-bottom path P'_x by replacing the portion of P_x which passes via n (connecting nodes n_{in} and n_{out}) by the portion of closed path P' that connects n_{in} and n_{out} (see Fig 7(a)). Since no nodes on P' are in S, the resulting top-bottom path P'_x does not pass via any node in S, contradicting the assumption that S is a top-bottom cutset, as required.

Proof of Claim 1. Starting from the initial closed path P satisfying properties (1) and (2), we repeatedly apply the following 'contraction' operation to P until we obtain the closed path P' satisfying (1),(2) and (3). The 'contraction' operation on P preserves properties (1) and (2), and can always be applied if P does not satisfy property (3). On the other hand, each 'contraction' operation reduces the area enclosed by P, whereas by property (1), S is contained within P, so P must always have a non-zero area. It follows that the 'contraction' operation can be applied at most a finite number of times to P before we obtain P' satisfying (3) as well as (1) and (2), as required.

We now define the contraction operation on P. Suppose that P does not satisfy property (3). Then there exists a sequence of 3 consecutive nodes $P(j-1), P(j), P(j+1)$ along the closed path P, where none of the nodes in $\mathcal{G}_{tri}(\ell, \ell)$ which are adjacent to $P(j)$ are in S_1. By symmetry considerations, it suffices to consider the 6 possibilities for $P(j-1), P(j), P(j+1)$ shown in Fig 8.

The 'contraction operation' for each of these cases consists of either adding a new node to P or removing a node from P, and is shown in Fig 8 using the following notation: a circled node indicates that the node is added to the path P, while a node enclosed with a triangle indicates that the node is removed from P. In each case, an arrow pointing away from the added/removed node shows a shaded area which is removed from the interior region of P by the contraction operation. Note that in each of the 6 cases, the interior region of P may be on either left-hand or right-hand side of the edges connecting $P(j-1), P(j), P(j+1)$, and accordingly, two contracting operations are shown for each of the 6 cases. Also observe that nodes added to P are always adjacent to $P(j)$ so that property (2) of P is preserved, and no nodes are contained inside the shaded areas removed from the interior region of P by the contraction operation, so property (1) of P is also preserved, as required. This completes the proof of the claim. □

Subcase 2: $r_{min} = 1, r_{max} = \ell$ **and either** $2 \leq c_{min}$ **or** $c_{max} \leq \ell - 1$**.** We may assume that $c_{max} \leq \ell - 1$, since the other case $c_{min} = 2$ is handled analogously by symmetry. Let $c_t \leq \ell - 1$ (resp. $c_b \leq \ell - 1$) denote the column index of the rightmost node of S on row 1 (resp. row ℓ). The 'C-shaped' path P_c in $\mathcal{G}_{tri}(\ell, \ell)$ connecting the 4 nodes $(1, c_t + 1)$,$(1, c_{max} + 1)$,$(\ell, c_{max} + 1)$ and $(\ell, c_b + 1)$, together with the left, bottom and top boundaries of $\mathcal{G}_{tri}(\ell, \ell)$ form a closed path P which encloses S_1. Thus P satisfies the properties (1) and (2) as in the previous subcase. Similarly to Claim 1 in the first subcase, we can repeatedly apply a contraction operation to P until we obtain P' satisfying (1), (2) and (3). The only differences in this subcase from the previous subcase are:

- In this subcase, the nodes $(1, c_t)$ and (ℓ, c_b) in P are also in P', since they are already adjacent to nodes of S_1.
- The final path P' connects the top node $(1, c_t)$ to the bottom node (ℓ, c_b) without passing via a node in S. Hence P' is a top-bottom path not passing via S, contradicting the assumption that S is a cutset.

Subcase 3: Exactly one of $r_{min} = 1$ **or** $r_{max} = \ell$ **holds and exactly one of** $c_{min} = 1$ **or** $c_{max} = \ell$ **holds.** We may assume that $r_{min} = c_{min} = 1$ and $r_{max}, c_{max} \leq \ell - 1$ since the other 3 cases are handled analogously by symmetry. The 'L-shaped' path P_L in $\mathcal{G}_{tri}(\ell, \ell)$ connecting the 3 nodes $(1, c_{max} + 1)$,$(r_{max} + 1, c_{max} + 1)$,$(r_{max} + 1, 1)$, together with the left and top boundaries of $\mathcal{G}_{tri}(\ell, \ell)$ form a closed path P which encloses S_1. Thus P satisfies the properties (1) and (2) as in the previous subcase. Similarly to Claim 1 in the first subcase, we can repeatedly apply a contraction operation to P until we obtain P' satisfying (1), (2) and (3). This path P' then contradicts the assumption that S is a cutset by a similar argument as in the first subcase, namely bypassing the portion of a top-bottom path passing via a node in S, with a portion along P' which does not contain nodes in S. Note, however, that in this subcase, only exit node n_{out} of P_x on P' is guaranteed to exist (see Fig 7(c) for an example), and P' connects a node on the top row to node n_{out}, before continuing along the original path P_x to the bottom row.

Subcase 4: $c_{min} = 1$ **and** $c_{max} = \ell$**.** This subcase implies (since S_1 is connected) that S_1 *properly* contains a right-left path, and hence is not a *minimal* cutset, contrary to assumption.

We conclude that in all cases, assuming that S is not a minimal right-left path but is a minimal top-bottom cutset, leads to a contradiction. This completes the proof of the 'only if' direction. The 'dual' claims of the Lemma (stated in parantheses) follow by symmetry of the graph $\mathcal{G}_{tri}(\ell, \ell)$ under a 90 degree rotation. This completes the proof of the Lemma. □

E A 1-Private n-Party Protocol

Theorem E.1 *Let $n \geq 3$ and G be a finite group. There exists a 1-private n-party protocol Π for computing $f_G(x_1, \ldots, x_n) = x_1 \cdots x_n$ which runs in $2n$ rounds and has total communication complexity $2n$ group elements.*

Proof. The protocol Π consists of the following $2n$ rounds (add/removing randomness phases):

- $P_1 \to P_2 : r_1 \cdot x_1$ for $r_1 \in_R G$.
- $P_2 \to P_3 : r_2 \cdot r_1 \cdot x_1 \cdot x_2$ for $r_2 \in_R G$, and so on up to round $n - 1$.
- ...
- $P_{n-1} \to P_n : r_{n-1} \cdots r_1 \cdot x_1 \cdots x_{n-1}$ for $r_{n-1} \in_R G$.
- $P_n \to P_{n-1} : r_{n-1} \cdots r_1 \cdot x_1 \cdots x_{n-1} \cdot x_n \cdot r_n$ for $r_n \in_R G$.
- $P_{n-1} \to P_{n-2} : r_{n-2} \cdots r_1 \cdot x_1 \cdots x_n \cdot r_n$, and so on up to round $2n - 2$.

- ...
- $P_1 \to P_n : x_1 \cdots x_n \cdot r_n$.
- $P_n \to \text{ALL} : x_1 \cdots x_n$.

Let $y = x_1 \cdots x_n$. The view of P_1 is $(r_1 \cdot y \cdot r_n, y)$, which is independent of $(x_2, \ldots, x_n)$ satisfying $x_1 \cdots x_n = y$. For $1 < i < n$, the view of P_i is $(r_{i-1} \cdots r_1) \cdot y \cdot x_i \cdot (x_{i+1} \cdots x_n), r_i \cdot (r_{i-1} \cdots r_1) \cdot y \cdot r_n, y)$, which is independent of $\{x_j\}_{j \neq i}$ satisfying $x_1 \cdots x_n = y$. The view of P_n is $((r_{n-1} \cdots r_1) \cdot y \cdot x_{n-1}, y \cdot r_n, y)$, which is independent of $(x_1, \ldots, x_{n-1})$ satisfying $x_1 \cdots x_n = y$. Thus Π is 1-private, completing the proof. □

F A 2-Private n-Party Protocol

Theorem F.1 *Let $n \geq 5$ and G be a finite group. There exists a 2-private n-party protocol Π for computing $f_G(x_1, \ldots, x_n) = x_1 \cdots x_n$ which runs in $n + 2$ rounds and has total communication complexity $O(n)$ group elements.*

Proof. We begin by presenting an n-party protocol $\mathsf{SnowBall}_n$ which is 2-private for $n \geq 8$. Using the results of the security analysis of this protocol we will then show how to 'patch' the cases $n \in \{5, 6, 7\}$ to make them 2-private by "thwarting" certain collusions with extra randomisation.

Protocol $\mathsf{SnowBall}_n$

- Round 1. $P_i \to P_{i+1} : r_{i,i+1 \bmod n}$ for each $i \in \{1, \ldots, n\}$, where $r_{i,i+1} \in_R G$.
- Round 2. $P_1 \to P_4 : M_{4,s} = r_1 x_1 r_{1,2}^{-1}$ and $P_2 \to P_4 : M_{4,c} = r_{1,2} x_2 r_{2,3}^{-1}$ (where $r_1 \in_R G$). P_4 multiplies to get $r_1 x_1 x_2 r_{2,3}^{-1}$.
- Round 3. $P_4 \to P_5 : M_{5,s} = r_1 x_1 x_2 r_{2,3}^{-1}$ and $P_3 \to P_5 : M_{5,c} = r_{2,3} x_3 r_{3,4}^{-1}$. P_5 multiplies to get $r_1 x_1 x_2 x_3 r_{3,4}^{-1}$.

  $\vdots$
- Round i $(3 \leq i \leq n-2)$. $P_{i+1} \to P_{i+2} : M_{i+2,s} = r_1 x_1 \cdots x_{i-1} r_{i-1,i}^{-1}$ and $P_i \to P_{i+2} : M_{i+2,c} = r_{i-1,i} x_i r_{i,i+1}^{-1}$. P_{i+2} multiplies to get $r_1 x_1 \cdots x_i r_{i,i+1}^{-1}$.

  $\vdots$
- Round $n-1$. $P_n \to P_2 : M_{2,s} = r_1 x_1 \cdots x_{n-2} r_{n-2,n-1}^{-1}$ and $P_{n-1} \to P_2 : M_{2,c} = r_{n-2,n-1} x_{n-1} r_{n-1,n}^{-1}$. P_2 multiplies to get $r_1 x_1 \cdots x_{n-1} r_{n-1,n}^{-1}$.
- Round n. $P_2 \to P_3 : M_{3,s} = r_1 x_1 \cdots x_{n-1} r_{n-1,n}^{-1}$ and $P_n \to P_3 : M_{2,c} = r_{n-1,n} x_n r_{n,1}^{-1}$. P_3 multiplies to get $r_1 x_1 \cdots x_n r_{n,1}$.
- Round $n+1$. $P_3 \to P_1$: $M_{1,s} = r_1 x_1 \cdots x_n r_{n,1}$. P_1 removes r_1 and $r_{n,1}$ to get $x_1 \cdots x_n$.
- Round $n+2$. $P_1 \to$ ALL: $y = x_1 \cdots x_n$.

The protocol is illustrated in Fig. 9.

In the following security analysis we divide all possible 2 player collusions $\{i, j\}_{1 \leq i < j \leq n}$ into several cases and subcases. For each of these cases, we write each of the received messages in the view of the colluding parties. For each of these viewed messages, we underline the randomness factor (if it exists) which is independent of the collusion's view so far (and hence allows the collusion to simulate the

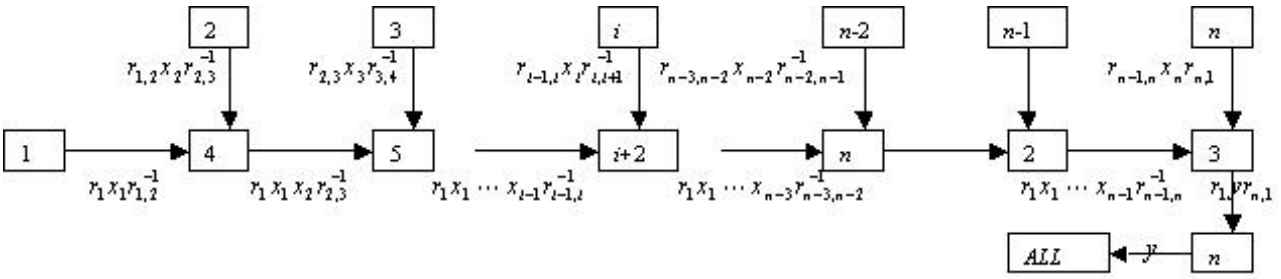

Figure 9: Protocol $\mathsf{SnowBall}_n$.

message by an independent random group element). In some cases such an independent randomness factor does not exist, but the received message can be simulated by the collusion by computing it from the other (simulatable) view of the collusion and the protocol output y. We indicate in brackets the reasoning and/or conditions for the simulation to work.

- Case $\{i, j\}$ with $4 \leq i < j \leq n$:
  - View(i): $M_{i,s} = r_1 x_1 \cdots x_{i-3} \underline{r_{i-3,i-2}^{-1}}$ $(i, j \notin \{i-3, i-2\})$, $M_{i,c} = r_{i-3,i-2} x_{i-2} \underline{r_{i-2,i-1}^{-1}}$ $(i, j \notin \{i-1, i-2\})$.
  - View(j): $M_{j,s} = \underline{r_1} x_1 \cdots x_{j-3} r_{j-3,j-2}^{-1}$ $(1 \notin \{i, j\})$, $M_{j,c} = \underline{r_{j-3,j-2}} x_{j-2} \underline{r_{j-2,j-1}^{-1}}$ (either $i \notin \{j-3, j-2\}$ or $i \notin \{j-2, j-1\}$ or $i = j-2$; in the last case $M_{j,c}$ is computable by P_i).
- Case $\{i, j\}$ with $i = 1$, $2 \leq j \leq n$:
  - View(1): $M_{1,s} = r_1 y r_{n,1}$ (computable by P_1)
  - Subcase $i = 1, j \geq 4$
    - View(j): $M_{j,s} = r_1 x_1 \cdots x_{j-3} \underline{r_{j-3,j-2}^{-1}}$ (either $1 \notin \{j-3, j-2\}$ or $j = 4$; in the last case $M_{j,s}$ is computable by P_1), $M_{j,c} = r_{j-3,j-2} x_{j-2} \underline{r_{j-2,j-1}^{-1}}$ $(1 \notin \{j-2, j-1\}$ since $j \geq 4)$.
  - Subcase $i = 1, j = 3$
    - View(3): $M_{3,s} = r_1 x_1 \cdots x_{n-1} \underline{r_{n-1,n}^{-1}}$ $(1, 3 \notin \{n-1, n\}$ since $n \geq 5)$, $M_{3,c} = r_{n-1,n} x_n r_{n,1}$ $(M_{3,c} = M_{3,s}^{-1} M_{1,s}$ computable by $\{P_1, P_3\})$.
  - Subcase $i = 1, j = 2$
    - View(2): $M_{2,s} = r_1 x_1 \cdots x_{n-2} \underline{r_{n-2,n-1}^{-1}}$ $(1, 2 \notin \{n-2, n-1\}$ since $n \geq 5)$, $M_{2,c} = r_{n-2,n-1} x_{n-1} \underline{r_{n-1,n}^{-1}}$ $(1, 2 \notin \{n-1, n\}$ since $n \geq 5)$.
- Case $\{i, j\}$ with $i = 2$, $3 \leq j \leq n$:
  - Subcase $i = 2, j \geq 6$
    - View(2): $M_{2,s} = \underline{r_1} x_1 \cdots x_{n-2} r_{n-2,n-1}^{-1}$ $(1 \notin \{2, j\})$, $M_{2,c} = \underline{r_{n-2,n-1}} x_{n-1} \underline{r_{n-1,n}^{-1}}$ (either $n-1, n \notin \{2, j\}$ or $n-2, n-1 \notin \{2, j\}$ or $j = n-1$; in the last case $M_{2,c}$ is computable by P_j).
    - View(j): $M_{j,s} = r_1 x_1 \cdots x_{j-3} \underline{r_{j-3,j-2}^{-1}}$ $(2 \notin \{j-3, j-2\}$ since $j \geq 6)$, $M_{j,c} = r_{j-3,j-2} x_{j-2} \underline{r_{j-2,j-1}^{-1}}$ $(2 \notin \{j-2, j-1\}$ since $j \geq 6)$.
  - Subcase $i = 2, j = 5$

* View(2): $M_{2,s} = r_1 x_1 \cdots x_{n-2} \underline{r^{-1}_{n-2,n-1}}$ ($n-2, n-1 \notin \{2,5\}$ **if** $\mathbf{n \geq 8}$), $M_{2,c} = r_{n-2,n-1} x_{n-1} \underline{r^{-1}_{n-1,n}}$ ($n-1, n \notin \{2,5\}$ **if** $\mathbf{n \geq 8}$).
* View(5): $M_{5,s} = \underline{r_1} x_1 x_2 r^{-1}_{2,3}$ ($1 \notin \{2,5\}$), $M_{5,c} = r_{2,3} x_3 \underline{r^{-1}_{3,4}}$ ($3, 4 \notin \{2,5\}$).

– Subcase $i = 2, j = 4$
* View(2): $M_{2,s} = r_1 x_1 \cdots x_{n-2} \underline{r^{-1}_{n-2,n-1}}$ ($n-2, n-1 \notin \{2,4\}$ **if** $\mathbf{n \geq 7}$), $M_{2,c} = r_{n-2,n-1} x_{n-1} \underline{r^{-1}_{n-1,n}}$ ($n-1, n \notin \{2,4\}$ **if** $\mathbf{n \geq 7}$).
* View(4): $M_{4,s} = \underline{r_1} x_1 r^{-1}_{1,2}$ ($1 \notin \{2,4\}$), $M_{4,c} = r_{1,2} x_2 \underline{r^{-1}_{2,3}}$ (computable by P_2).

– Subcase $i = 2, j = 3$
* View(2): $M_{2,s} = \underline{r_1} x_1 \cdots x_{n-2} r^{-1}_{n-2,n-1}$ ($1 \notin \{2,3\}$), $M_{2,c} = \underline{r_{n-2,n-1}} x_{n-1} \underline{r^{-1}_{n-1,n}}$ (either $n-1, n \notin \{2,3\}$ or $n-2, n-1 \notin \{2,3\}$ since $n \geq 5$).
* View(3): $M_{3,s} = r_1 x_1 \cdots x_{n-1} r^{-1}_{n-1,n}$ ($M_{3,s} = M_{2,s} M_{2,c}$ computable by P_2), $M_{3,c} = r_{n-1,n} x_n \underline{r_{n,1}}$ ($1, n \notin \{2,3\}$ since $n \geq 5$).

- Case $\{i, j\}$ with $i = 3$, $4 \leq j \leq n$:

– Subcase $i = 3, j \geq 7$
* View(3): $M_{3,s} = \underline{r_1} x_1 \cdots x_{n-1} r^{-1}_{n-1,n}$ ($1 \notin \{3, j\}$), $M_{3,c} = r_{n-1,n} x_n \underline{r_{n,1}}$ (either $1, n \notin \{3, j\}$ or $j = n$; in the last case $M_{3,c}$ is computable by P_j).
* View(j): $M_{j,s} = r_1 x_1 \cdots x_{j-3} \underline{r^{-1}_{j-3,j-2}}$ ($3 \notin \{j-3, j-2\}$ since $j \geq 7$), $M_{j,c} = r_{j-3,j-2} x_{j-2} \underline{r^{-1}_{j-2,j-1}}$ ($3 \notin \{j-2, j-1\}$ since $j \geq 7$).

– Subcase $i = 3, j = 6$
* View(3): $M_{3,s} = r_1 x_1 \cdots x_{n-1} \underline{r^{-1}_{n-1,n}}$ ($n-1, n \notin \{3,6\}$ **if** $\mathbf{n \geq 8}$), $M_{3,c} = r_{n-1,n} x_n \underline{r_{n,1}}$ ($1, n \notin \{3,6\}$ **if** $\mathbf{n \geq 8}$).
* View(6): $M_{6,s} = \underline{r_1} x_1 x_2 x_3 r^{-1}_{3,4}$ ($1 \notin \{3,6\}$), $M_{6,c} = r_{3,4} x_4 \underline{r^{-1}_{4,5}}$ ($4, 5 \notin \{3,6\}$).

– Subcase $i = 3, j = 5$
* View(3): $M_{3,s} = r_1 x_1 \cdots x_{n-1} \underline{r^{-1}_{n-1,n}}$ ($n-1, n \notin \{3,5\}$ **if** $\mathbf{n \geq 7}$), $M_{3,c} = r_{n-1,n} x_n \underline{r_{n,1}}$ ($1, n \notin \{3,5\}$ **if** $\mathbf{n \geq 7}$).
* View(5): $M_{5,s} = \underline{r_1} x_1 x_2 r^{-1}_{2,3}$ ($1 \notin \{3,5\}$), $M_{5,c} = r_{2,3} x_3 r^{-1}_{3,4}$ (computable by P_3).

– Subcase $i = 3, j = 4$
* View(3): $M_{3,s} = r_1 x_1 \cdots x_{n-1} \underline{r^{-1}_{n-1,n}}$ (either $n-1, n \notin \{3,4\}$ or $n = 5$; in the last case $M_{3,s} = M_{4,s} M_{4,c} r_{2,3} x_3 x_4 r^{-1}_{4,5}$ is computable by P_3, P_4).
* View(4): $M_{4,s} = \underline{r_1} x_1 r^{-1}_{1,2}$ ($1 \notin \{3,4\}$), $M_{4,c} = \underline{r_{1,2}} x_2 r^{-1}_{2,3}$ ($1, 2 \notin \{3,4\}$).

From the above analysis, we conclude that $\mathsf{SnowBall}_n$ is 2-private for $n \geq 8$. We can also see that the insecure collusions for the cases $n \in \{5, 6, 7\}$ are as follows:

- $n = 7$: $\{2,5\}$, $\{3,6\}$
- $n = 6$: $\{2,4\}$, $\{2,5\}$, $\{3,5\}$, $\{3,6\}$
- $n = 5$: $\{2,4\}$, $\{2,5\}$, $\{3,5\}$

Patching $\mathsf{SnowBall}_n$ *for the cases* $n \in \{5, 6, 7\}$. The above collusions $\{i, j\}$ are insecure for the stated values of n because the 'state' messages $M_{i,s}$ and $M_{j,s}$ received by $\{i, j\}$ contain only one randomness value r_1 which is not known to $\{i, j\}$. This leads to an attack where one of the messages $M_{i,s}, M_{j,s}$ is used to solve for r_1 and the other to get information on player secrets and hence break

the protocol. Accordingly, the idea for thwarting each of the above collusions $\{i, j\}$ is to generate a new independent random group element $\widehat{r}_{i',j'}$ shared between a pair of players $\{i', j'\}$, where i', j' are chosen such that:

- i', j' are not in the collusion $\{i, j\}$.
- The subsequence of players between i' and j' along the protocol's multiplier player sequence (i.e. the sequence $(1, 4, 5, \ldots, n, 2, 3)$ of players which multiply pairs of received group elements) contains exactly *one* of the collusion players $\{i, j\}$, say player i.

If these conditions are met, then the protocol can be 'patched' as follows (exploiting the fact that the 'multiplier' players in the original protocol only multiply elements on the *right* of the 'state' element): the new random element $\widehat{r}_{i',j'}$ (exchanged by players i',j' in round 1) is multiplied by player i' on the left of the state element (we assume that i' occurs before j' along the protocol's multiplier player sequence), and this randomness is later removed by player j' when the state element reaches it. Thanks to the second condition above, the result of this patch is that the new random element $\widehat{r}_{i',j'}$ appears as the leftmost factor in the state element $M_{i,s}$ received by player i (which is thus simulatable by $\{i, j\}$ thanks to independence of $\widehat{r}_{i',j'}$ from the view of $\{i, j\}$), but not in the state element $M_{j,s}$ received by player j (which is thus simulatable by $\{i, j\}$ thanks to independence of r_1 from the view of $\{i, j\}$). Hence the patched protocol becomes secure against the collusion $\{i, j\}$ (without becoming insecure against any other collusions).

In Fig. 10, we illustrate for each of the cases $n \in \{5, 6, 7\}$ the insecure collusions for the unpatched $\mathsf{SnowBall}_n$ protocol, and the choices for $\{i', j'\}$ used to thwart those collusions in the patched protocol as explained above. We remark that for the case $n = 5$, there is one collusion, namely $\{2, 5\}$, which cannot be thwarted by the above method, because players 2 and 5 are *adjacent* in the protocol's multiplier player sequence (so i',j' satisfying the above conditions do not exist). To thwart this collusion, we use in the patched protocol a shared randomness $\widehat{r}_{3,4}$ between players $\{3, 4\}$. The player 4 multiplies $\widehat{r}_{3,4}$ on the *right* of his 'contribution' message $M_{2,c}$ to player 2. The randomness $\widehat{r}_{3,4}$ is removed from the state element by the following player 3 before multiplying the contribution of player 5 on the right. This patch suffices to thwart the collusion $\{2, 5\}$.

For clarity, we give below the details of the abovementioned patches shown in Fig 10 for the $n = 5$ protocol, along with the security analysis which proves that they thwart the insecure collusions of the original protocol.

The patches for protocol $\mathsf{SnowBall}_5$ are the following:

- Round 1 additions. $P_1 \to P_5 : \widehat{r}_{1,5}$, $P_1 \to P_2 : \widehat{r}_{1,2}$, $P_3 \to P_4 : \widehat{r}_{3,4}$
- Round 2 mods. $P_1 \to P_4 : M_{4,s} = \widehat{r}_{1,5} r_1 x_1 r_{1,2}^{-1}$
- Round 4 mods. $P_5 \to P_2 : M_{2,s} = r_1 x_1 x_2 x_3 r_{3,4}^{-1}$ (P_5 removed $\widehat{r}_{1,5}$), $P_4 \to P_2 : M_{2,c} = r_{3,4} x_4 r_{4,5}^{-1} \widehat{r}_{3,4}$
- Round 5 mods. $P_2 \to P_3 : M_{3,s} = \widehat{r}_{1,2} r_1 x_1 x_2 x_3 x_4 r_{4,5}^{-1} \widehat{r}_{3,4}$, $P_5 \to P_3 : r_{4,5} x_5 r_{5,1}$. P_3 removes $\widehat{r}_{3,4}$ from $M_{3,s}$ and multiplies to get $\widehat{r}_{1,2} r_1 x_1 x_2 x_3 x_4 x_5 r_{5,1}$.
- Round 6 mods. $P_3 \to P_1 : M_{1,s} = \widehat{r}_{1,2} r_1 y r_{5,1}$. P_1 removes $\widehat{r}_{1,2} r_1$ and $r_{5,1}$ to get y.

Let us now formally verify that the patched protocol is indeed secure against the collusions $\{2, 4\}$, $\{2, 5\}$, $\{3, 5\}$:

- Collusion $\{2, 4\}$
  - View(2): $M_{2,s} = \underline{r_1} x_1 x_2 x_3 r_{3,4}^{-1}$ ($1 \notin \{2, 4\}$), $M_{2,c} = r_{3,4} x_4 r_{4,5}^{-1} \widehat{r}_{3,4}$ (computable by P_4).
  - View(4): $M_{4,s} = \underline{\widehat{r}_{1,5}} r_1 x_1 r_{1,2}^{-1}$ ($1, 5 \notin \{2, 4\}$), $M_{4,c} = r_{1,2} x_2 r_{2,3}^{-1}$ (computable by P_2).

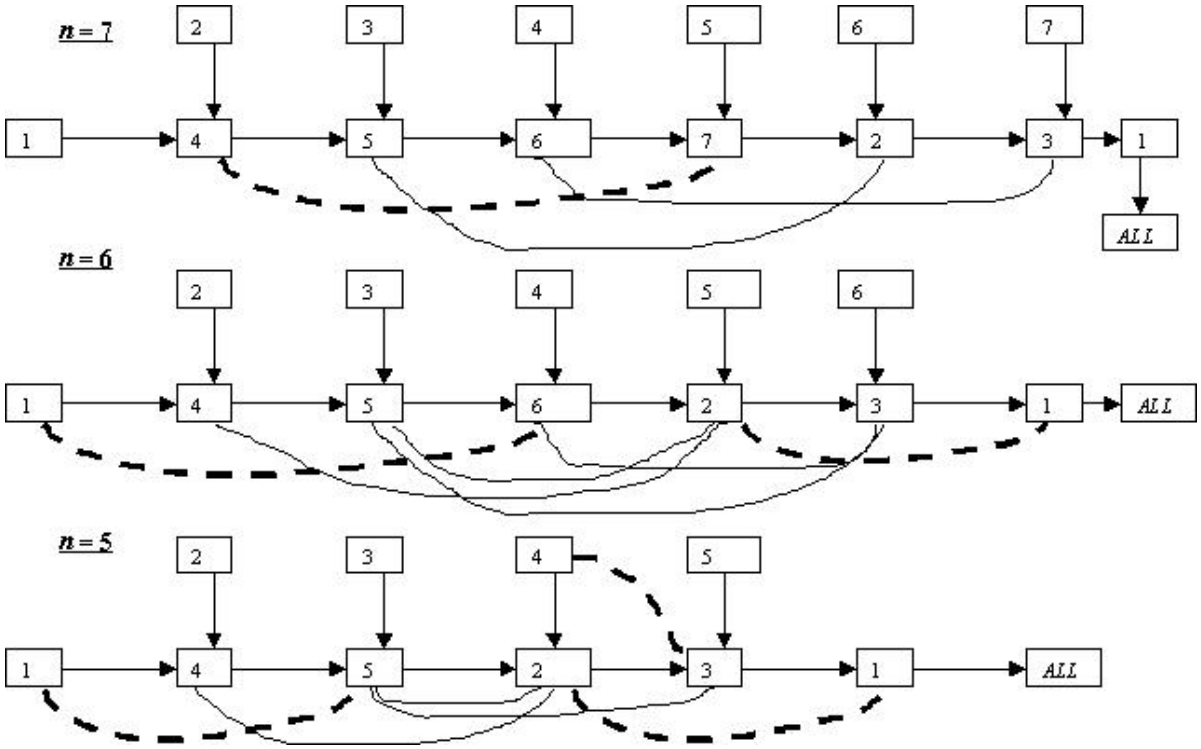

Figure 10: Patches for $\mathsf{SnowBall}_n$ cases $n \in \{5, 6, 7\}$. The solid lines indicate the insecure collusions $\{i, j\}$ in the original protocols. The dashed lines indicate the pairs of players $\{i', j'\}$ chosen to share a new randomness value $r_{i',j'}$ in the patched protocols in order to thwart the insecure collusions, as explained in the text.

- Collusion $\{2, 5\}$
  - View(2): $M_{2,s} = r_1 x_1 x_2 x_3 r_{3,4}^{-1}$ ($M_{2,s} = \widehat{r}_{1,5}^{-1} M_{5,s} M_{5,c}$ computable by P_5), $M_{2,c} = r_{3,4} x_4 r_{4,5}^{-1} \underline{\widehat{r}_{3,4}}$ ($3, 4 \notin \{2, 5\}$).
  - View(5): $M_{5,s} = \widehat{r}_{1,5} \underline{r_1} x_1 x_2 r_{2,3}^{-1}$ ($1 \notin \{2, 5\}$), $M_{5,c} = r_{2,3} x_3 \underline{r_{3,4}^{-1}}$ ($3, 4 \notin \{2, 5\}$).
- Collusion $\{3, 5\}$
  - View(3): $M_{3,s} = \underline{\widehat{r}_{1,2}} r_1 x_1 x_2 x_3 x_4 r_{4,5}^{-1} \widehat{r}_{3,4}$ ($1, 2 \notin \{3, 5\}$), $M_{3,c} = r_{4,5} x_5 r_{5,1}$ (computable by P_5).
  - View(5): $M_{5,s} = \widehat{r}_{1,5} \underline{r_1} x_1 x_2 r_{2,3}^{-1}$ ($1 \notin \{3, 5\}$), $M_{5,c} = r_{2,3} x_3 r_{3,4}^{-1}$ (computable by P_3).

We conclude that the patched $\mathsf{SnowBall}_5$ protocol is 2-private. We leave it to the reader to verify the security of the patches described in Fig. 10 for the cases $n \in \{6, 7\}$. This completes the proof of the theorem. □