J. Cryptol. (2012) 25: 557–600
DOI: 10.1007/s00145-011-9104-3

Journal of
CRYPTOLOGY

Graph Coloring Applied to Secure Computation in Non-Abelian Groups*

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Josef Pieprzyk and Ron Steinfeld
Department of Computing, Macquarie University, Sydney, Australia

Xiaoming Sun and Christophe Tartary
Institute for Interdisciplinary Information Sciences, Institute for Theoretical Computer Science, Tsinghua University, Beijing, People's Republic of China
ctartary@mail.tsinghua.edu.cn

Huaxiong Wang
School of Physical and Mathematical Sciences, Division of Mathematical Sciences, Nanyang Technological University, Singapore, Singapore

Andrew Chi-Chih Yao
Institute for Interdisciplinary Information Sciences, Institute for Theoretical Computer Science, Tsinghua University, Beijing, People's Republic of China

Communicated by Ivan Damgard

Received 22 August 2010
Online publication 6 September 2011

Abstract. We study the natural problem of secure n-party computation (in the computationally unbounded attack model) of circuits over an arbitrary finite non-Abelian group $(G, \cdot)$, which we call G-circuits. Besides its intrinsic interest, this problem is also motivating by a completeness result of Barrington, stating that such protocols can be applied for general secure computation of *arbitrary* functions. For flexibility, we are interested in protocols which only require *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 investigations focus on the passive adversarial model, where up to t of the n participating parties are corrupted.

Our results are as follows. We initiate a novel approach for the construction of black-box protocols for G-circuits based on k-of-k threshold secret-sharing schemes, which are efficiently implementable over any black-box (non-Abelian) group G. We reduce the problem of constructing such protocols to a combinatorial coloring problem in planar graphs. We then give three constructions for such colorings. Our first approach leads to a protocol with optimal resilience $t < n/2$, but it requires exponen-

* This paper is an extended version of [13,28].

tial communication complexity $O(\binom{2t+1}{t}^2 \cdot N_g)$ group elements and round complexity $O(\binom{2t+1}{t} \cdot N_g)$, for a G-circuit of size N_g. Nonetheless, using this coloring recursively, we obtain another protocol to t-privately compute G-circuits with communication complexity $\mathcal{P}oly(n) \cdot N_g$ for any $t \in O(n^{1-\epsilon})$ where ϵ is *any* positive constant. For our third protocol, there is a probability δ (which can be made arbitrarily small) for the coloring to be flawed in term of security, in contrast to the first two techniques, where the colorings are always secure (we call this protocol probabilistic, and those earlier protocols deterministic). This third protocol achieves optimal resilience $t < n/2$. It has communication complexity $O(n^{5.056}(n + \log \delta^{-1})^2 \cdot N_g)$ and the number of rounds is $O(n^{2.528} \cdot (n + \log \delta^{-1}) \cdot N_g)$.

Key words. Multiparty computation, Graph coloring, Non-Abelian group, Black-box operations, Planar graph, Percolation theory, Word problem.

1. Introduction

Background Groups form a natural mathematical structure for cryptography. In particular, the most popular public-key encryption schemes nowadays (RSA [25] and Diffie-Hellman/ElGamal [15,16]) operate in Abelian groups. The idea of using non-Abelian groups in cryptography dates back to 1985 where [29] described a public-key protocol based on the hardness of the word problem for finitely presented groups. The past few years saw an increased interest in the cryptographic use of non-Abelian groups [22–24]. Moreover, a result due to Barrington [3] shows that computation in the non-Abelian symmetric group S_5 is *complete* in the sense that it can be used to perform computation of *arbitrary* functions.

Motivated by the above, we study the natural problem of secure MultiParty Computation (MPC), in the computationally unbounded attack model, of circuits over an arbitrary finite group $(G, \cdot)$ consisting of gates performing the group operation of G. We call such circuits G-circuits. For flexibility, we are interested in protocols for G-circuits which require only *black-box* access to the group G (i.e. the only computations performed by the players during 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 n-party protocol exists for G-circuits which is t-private, i.e. secure against t passive parties (also called semi-honest parties) for any $t < n$ and has communication complexity $O(n^2)$ group elements. This protocol is based on the concept of a *homomorphic* secret-sharing scheme [5]; for a group $(G, \cdot)$, these schemes have the property that, if $\vec{v} = (v_1, \ldots, v_n) \in G^n$ is a share vector for a secret $s \in G$, and $\vec{w} = (w_1, \ldots, w_n) \in G^n$ is a share vector for a secret $\tilde{s} \in G$, then $\vec{u} = (v_1 \cdot w_1, \ldots, v_n \cdot w_n) \in G^n$ is a share vector for the secret $r = s \cdot \tilde{s} \in G$. For example, given an n-of-n homomorphic secret-sharing scheme, a protocol for shared computation of a product $x_1 \cdots x_n$, where x_i is held by P_i for $i = 1, \ldots, n$ works as follows. In the first round, for each i, party P_i computes a share vector $\vec{v}_i = (v_{i,1}, \ldots, v_{i,n})$ for its input x_i, and for each j, sends the jth share $v_{i,j}$ to party P_j. In the second round, for each j, party P_j computes the product $u_j = v_{1,j} \cdots v_{n,j}$ of the shares it received in the first round, and broadcasts u_j to all parties. By the homomorphic property, $\vec{u} = (u_1, \ldots, u_n)$ is a share vector for the desired value $x_1 \cdots x_n$, which all parties can then compute.

Unfortunately, when $(G, \cdot)$ is non-Abelian, the above construction of a homomorphic secret-sharing scheme fails, and in fact it is known that no homomorphic construction exists [17]. Thus, the above 2-round protocol no longer applies. Moreover, to our knowledge, when $(G, \cdot)$ is non-Abelian, no constructions of black-box protocols for G-circuits have been designed until now. Consequently, to construct a t-private protocol for G-circuits over 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 work only considers the passive attack model. We obtain the following results. We initiate a novel approach for the construction of black-box MPC protocols for G-circuits 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 field and its generalization to rings). We reduce the problem of constructing such protocols to a combinatorial coloring problem in planar graphs. Our notion of 't-reliable n-coloring' is closely related to the notion of set separability defined in [12], 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 colors and the adversary controls all nodes with colors belonging to a t-color subset I. However, in our paper, the behavior of colored nodes is different from [12]. Indeed, our coloring is used for MPC both at the global level and at the node level (each node performs a sharing) while, in [12], colored nodes are only used to transmit data. In our context, we need to ensure a specific graph connectivity property for all t-sets of adversaries. We then give three constructions for such graph colorings. Our first coloring construction gives a protocol with optimal resilience $t < n/2$, but it has exponential communication complexity $O(\binom{2t+1}{t}^2 \cdot N_g)$ group elements and round complexity $O(\binom{2t+1}{t} \cdot N_g)$, for a G-circuit of size N_g (this construction also easily generalizes to general Q^2-adversary structures $\mathcal{A}$ as defined in [20], giving communication complexity $O(|\mathcal{A}|^2 \cdot N_g)$ group elements). We show how to use it recursively to construct a protocol with polynomial communication complexity for any $t \in O(n^{1-\epsilon})$ where ϵ is any positive constant. Our last coloring construction is probabilistic. Using arguments from percolation theory, we demonstrate that our third construction gives a protocol with optimal resilience ($t < n/2$) while having polynomial communication complexity (as low as $O(t^2 \cdot N_g)$ group elements when $t \leq n/(2+\epsilon)$ for any constant $\epsilon > 0$). The reader can find a detailed table containing the different parameters for our protocols in Table 1 located at the end of this paper in Sect. 4.

Differences with [13,28] Our current paper presents the following improvements of our earlier work.

1. We generalize our approach to MPC over non-Abelian groups by providing protocols allowing the computation of any G-circuit. In [13,28], we restricted ourselves to the case of the n-product function $f_G(x_1, \ldots, x_n) = x_1 \cdots x_n$.
2. We refine the percolation analysis performed in [28] to demonstrate the feasibility of probabilistic MPC for any $t < n/2$ with polynomial communication complexity.

3. We perform an explicit analysis of the round complexity for our different protocols.
4. We provide a set of graphs exhibiting a t-reliable $(2t+1)$-coloring for some small values of t. Despite the fact that we were not able to generalize them, we believe that they might show some light as to how to obtain general constructions.

Related Work There are two known non black-box methods for constructing a t-private protocol for G-circuits for any $t < n/2$. They are both based on Shamir's t-of-n threshold secret-sharing scheme [26] and its generalizations.

The first method [4,7,18] requires representing the G-circuit C 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 $O(t^2 \log t \cdot N_{\mathrm{AND}}(C))$ bits, where $N_{\mathrm{AND}}(C)$ denotes the number of AND gates in the Boolean AND/NOT circuit for computing C. Thus, this protocol is efficient only for very small groups G, for which $N_{\mathrm{AND}}(C)$ is manageable. Improvements can be found in [9,11] where the cost is reduced to $O(d(\log d)\mathcal{P}oly(n) + N_{\mathrm{AND}}(C)\mathcal{P}oly(\log n))$ (where d represents the depth of the circuit) and $O(n(\log n)N_{\mathrm{AND}}(C))$, respectively. However, these values still depend linearly on $N_{\mathrm{AND}}(C)$.

The second method [8] (see also [2] for earlier work) requires representing a G-circuit 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 $O(t^2 \log t \cdot N_M(C) \cdot \ell(R))$ bits, where $N_M(C)$ is the number of multiplication operations in the implementation of the G-circuit C over R and $\ell(R) \geq \log |R|$ denotes the number of bits needed for representing elements of R. If we 'embed' the group G in the ring $R = R(G)$, so that R inherits the multiplication operation of G, then $N_M(C) = n-1$, and hence the protocol from [8] has total communication complexity $O(N_g t^2 \log t \cdot \ell(R(G)))$ bits (where N_g denotes the size of the G-circuit C), compared to $O(N_g t^2 \cdot \ell(G))$ bits for our probabilistic protocol, where $\ell(G) \geq \log |G|$ is the representation length of elements of G. Hence, the communication complexity of our third protocol for C is smaller than the one from [8] by a factor $\Theta(\frac{\ell(R(G))}{\ell(G)} \cdot \log t)$. However, for this generic choice of $R(G)$, we have $\ell(R(G)) = |G|$. Thus, 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] as in 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 $O(N_g t^2 \cdot \ell(R(G)))$. A generalization of the technique from [11] to rings might give another way to save a factor $\Theta(\log t)$ for circuits of low depth.

We wish to emphasize a fundamental difference between our protocols and all previous MPC constructions discussed above. Namely, previous protocols achieve security against t colluding parties by using a $(t+1)$-of-n *threshold* secret-sharing scheme to distribute the protocol inputs and intermediate protocol values. The focus is on designing special threshold secret-sharing schemes that allow protocols for computations on shares. In contrast, our protocols use only a very simple k-of-k secret-sharing scheme with no special properties, and we achieve the security against t colluding parties by designing a special protocol communication graph. Thus, our approach shifts the design focus from secret-sharing schemes to communication graphs.

Organization The rest of the paper is organized as follows. Section 2 presents a sequence of reductions from the problem of designing secure MPC protocols to a specific graph coloring problem. Following some preliminaries in Sect. 2.1, Sect. 2.2 reviews Barrington's reduction of general Boolean circuit computation to computation of circuits over the non-Abelian symmetric group S_5. This shows that our protocols, applied over S_5, can be used for general secure MPC. Next, in Sect. 2.3, we reduce the problem of secure MPC over a general group G to the *shared 2-product* problem of multiplying two group elements, where at the start of the protocol, the parties hold shares of the two elements to be multiplied, and at its end, they hold shares of the product. Then, in Sect. 2.4, we reduce the shared 2-product problem to a graph coloring problem. In Sect. 3, we show a relaxation of the coloring conditions and then we present constructions meeting these new conditions. More specifically, Sect. 3.3 contains two deterministic coloring constructions, while Sect. 3.4 studies a probabilistic coloring technique allowing more efficient protocols at the cost of an arbitrarily small probability of insecurity. The last section of this paper concludes with some open problems. Appendices A–E contain several proofs of results referred to in the main text.

2. From Multiparty Computation to Graph Coloring

In this section, we propose a two-step reduction from the problem of designing protocols for t-privately computing G-circuits to the issue of obtaining a particular coloring for some specific families of planar graphs.

2.1. *Preliminaries*

2.1.1. *Security Model*

We recall the definition of secure MPC in the passive (semi-honest), computationally unbounded attack model, restricted to deterministic symmetric functionalities and perfect emulation [18]. Let $[n]$ denote the set $\{1, \ldots, n\}$.

Definition 1. Let $f : (\{0,1\}^*)^n \to \{0,1\}^*$ denote an n-input, single-output function, and let $\prod$ be an n-party protocol for computing f. We denote the party input sequence by $\vec{x} = (x_1, \ldots, x_n)$, the joint protocol view of parties in subset $I \subseteq [n]$ by $\mathsf{VIEW}_I^{\prod}(\vec{x})$, and the protocol output by $\mathsf{OUT}^{\prod}(\vec{x})$. For $0 < t < n$, we say that $\prod$ 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 $\vec{x} \in (\{0,1\}^*)^n$, the random variables

$$\left\langle \mathsf{S}\big(I, \vec{x}_I, f(\vec{x})\big), f(\vec{x}) \right\rangle \quad \text{and} \quad \left\langle \mathsf{VIEW}_I^{\prod}(\vec{x}), \mathsf{OUT}^{\prod}(\vec{x}) \right\rangle$$

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

2.1.2. *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},
- 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 as described in Proposition 1. We use this k-of-k scheme as a fundamental building block in our protocols. The following proposition is easy to prove.

Proposition 1. *Fix $x \in G$ and integers k and $j \in [k]$, and suppose we create an 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.*

2.2. *Reducing General Multiparty Computation to G-Circuit Computation*

As we explained in the introduction, a strong motivation for our study of MPC over non-Abelian groups G is the fact that it is *complete* for general MPC, i.e. it provides a new method of performing secure MPC of *arbitrary* functions. In this section, 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. Our protocol can then be applied over S_5 to securely compute the S_5-circuit C' and hence the Boolean circuit C.

In the following, for a group G, we define an m-input 1-output G-circuit C as a circuit (directed acyclic graph) with m 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).

Since we work over general groups that may be non-Abelian, the order of inputs to each Mult gate is important; accordingly, we assume that for each Mult node in the circuit C, one of its incoming edges is labeled by 'x', indicating that the corresponding input is to be multiplied on the left, and we call it the 'x-input' edge. We call the remaining incoming edge the 'y-input' edge. For each node, we allow arbitrarily many

outgoing edges, which all carry the same value (the gate's output value), allowing arbitrary 'fan out'.

We denote by $f_C : G^m \to G$ 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 $g(x_1, \ldots, x_m) = \phi_\sigma^{-1}(f_C(\phi_\sigma(x_1), \ldots, \phi_\sigma(x_m)))$ for all $(x_1, \ldots, x_m) \in \{0,1\}^m$.

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

Theorem 1 (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 $z = \mathrm{AND}(x, y)$ 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) $x_{\sigma_1^{-1}}, y_{\sigma_2}, y_{\sigma_2^{-1}}$.
- 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 $z = \mathrm{AND}(x, y)$ 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 $\mathrm{NOT}(x)_{\sigma_1^{-1}} = x_{\sigma_1} \cdot \sigma_1^{-1}$. 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. □

Remark 1. It is well known [4] that for general secure MPC in the passive setting, an honest majority ($t < n/2$) is necessary. Consequently, the above result of Barrington immediately implies that an honest majority is also necessary for MPC of G-circuits for arbitrary groups G. An alternative proof of this necessary condition, not relying on Barrington's result, can be found in [13].

2.3. *Construction of a G-Circuit Protocol from a Shared 2-Product Protocol*

In this section, we begin our study of MPC over G-circuits, by reducing the problem of constructing a n-party, t-private protocol for computing a G-circuit, to the problem of constructing a subprotocol for the *Shared* 2-Product function $f'_G(x, y) = x \cdot y$, where inputs x, y and output $z = x \cdot y$ are shared among the parties. We define for the latter subprotocol a so-called *strong* t-privacy definition, which will be needed later to prove the (standard) t-privacy of the full G-circuit protocol. 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. Note that although the latter requirement output simulation is stronger than the standard t-privacy requirement, the simulator here is given additional help as input, namely the values of all input shares except one share for x and one share for y. In this sense, strong t-privacy is a weaker requirement than the standard t-privacy: indeed, a shared 2-product subprotocol satisfying strong t-privacy may not satisfy the standard t-privacy definition (e.g. one of the input shares not known to the adversary may be revealed to the adversary by the internal subprotocol view; this violates the standard t-privacy, but not necessarily the strong t-privacy).

Definition 2 (n-Party Shared 2-Product Subprotocol). A n-party shared 2-product subprotocol $\prod_S$ 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 $\prod_S$ 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 $\prod_S$ achieves *strong t-privacy* if there exists a probabilistic simulator algorithm $\mathsf{S}_{\prod_S}$ 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

$$\left\langle \mathsf{S}_{\prod_S}\left(I, \left\{\boldsymbol{s}_x(j)\right\}_{j \in [\ell] \setminus \{j_x^*\}}, \left\{\boldsymbol{s}_y(j)\right\}_{j \in [\ell] \setminus \{j_y^*\}}\right)\right\rangle \quad \text{and}$$

$$\left\langle \mathsf{VIEW}_I^{\prod_S}(\boldsymbol{s}_x, \boldsymbol{s}_y), \left\{s_z(j)\right\}_{j \in [\ell] \setminus \{j_z^*\}}\right\rangle$$

are identically distributed (over the random coins of $\prod_S$). Here, $\mathsf{VIEW}_I^{\prod_S}(\boldsymbol{s}_x, \boldsymbol{s}_y)$ denotes the view of I in subprotocol $\prod_S$ 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 $\prod_S$ 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 $\prod_S$ achieves *symmetric strong t-privacy*.

Remark 2. 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 3. The adversary view simulator $\mathsf{S}_{\prod_S}$ 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 $\prod_S$ which achieves symmetric strong t-privacy, we may use a pair of subprotocols $\prod_S^{(x)}, \prod_S^{(y)}$ which are x-preserving and y-preserving, respectively, and are *compatible* in the following sense.

Definition 3 (Compatible Subprotocols). Let $\prod_S^{(x)}$ and $\prod_S^{(y)}$ denote two shared 2-Product subprotocols which satisfy x-preserving strong t-privacy and y-preserving strong t-privacy, respectively. We say $\prod_S^{(x)}$ and $\prod_S^{(y)}$ are *compatible* if:

- $\prod_S^{(x)}$ and $\prod_S^{(y)}$ have the same share ownership functions $\mathcal{O}_x, \mathcal{O}_y$.
- For each collusion $I \subset [n]$ with $|I| \leq t$, $\prod_S^{(x)}$ and $\prod_S^{(y)}$ have the same j_x^* index and the same j_y^* index (defined as in Definition 2).

The idea is that if $\prod_S^{(x)}$ and $\prod_S^{(y)}$ are compatible, we can use the output shares of a $\prod_S^{(x)}$ run as x-input shares to a following $\prod_S^{(y)}$ 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 compatibility, we can use the output of the first simulator as input to the second simulator. Note that if $\prod_S$ satisfies symmetric strong t-privacy then $\prod_S$ is compatible with itself, i.e. one can take $\prod_S^{(x)} = \prod_S^{(y)} = \prod_S$. 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.

Next, for an arbitrary m-input G-circuit C, we construct a protocol $\prod(C, \prod_S^{(x)}, \prod_S^{(y)})$ for private computation of C, using a pair of compatible shared 2-product subprotocols $\prod_S^{(x)}$ and $\prod_S^{(y)}$ (see Definition 3). We assume that $\prod_S^{(x)}$ (resp. $\prod_S^{(y)}$) satisfy

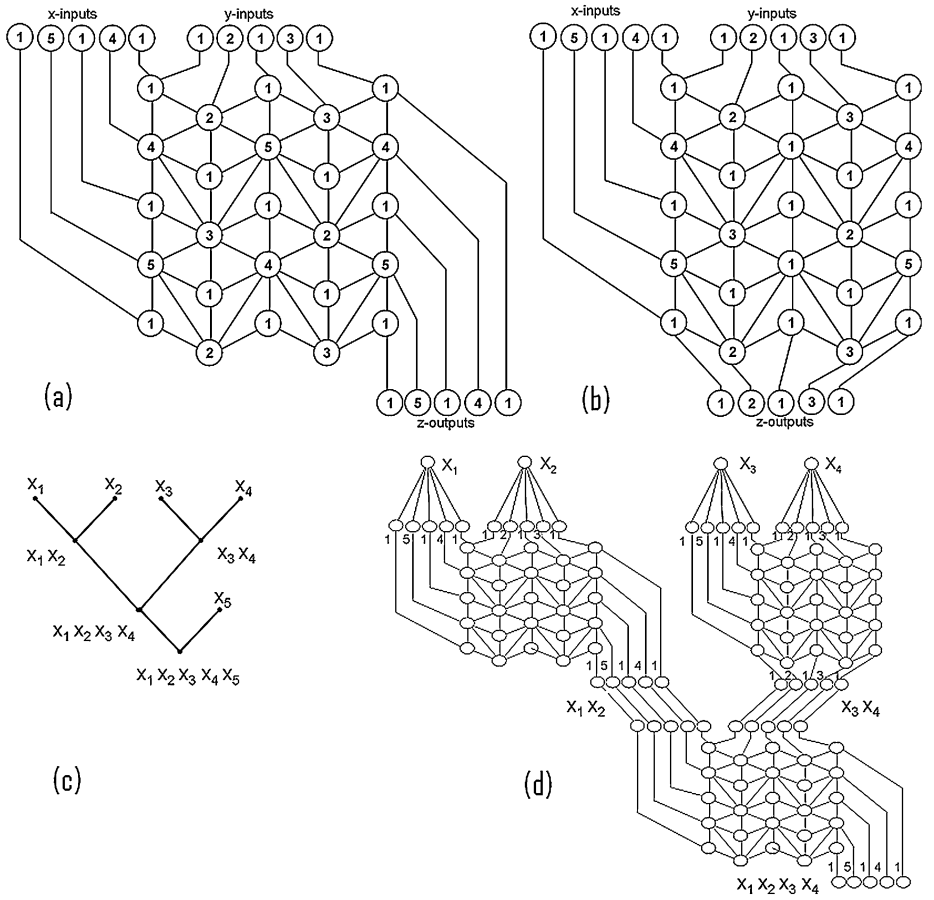


Fig. 1. (**a**) Example of a 5-party shared 2-product subprotocol $\prod_S^{(x)}$ satisfying x-preserving strong 2-privacy, with sharing parameter $\ell = 5$. The source nodes on the top row are labeled with indices of parties holding x-input and y-input shares, according to share ownership functions $\mathcal{O}_x, \mathcal{O}_y$. 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 Sect. 2.4.2 for more details). (**b**) Example of a 5-party shared 2-product subprotocol $\prod_S^{(y)}$ satisfying y-preserving strong 2-privacy. Note it is identical to $\prod_S^{(x)}$, except that outputs are taken from a different set of nodes. Also note that $\mathcal{O}_z = \mathcal{O}_y$. (**c**) Example of a G-circuit C for function $f_C(x_1, \ldots, x_5) = x_1x_2x_3x_4x_5$ with $m = n = 5$. Input 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 circuit C in (**c**)) in the protocol $\prod(C, \prod_S^{(x)}, \prod_S^{(y)})$ for f_C constructed from circuit C in (**c**) and subprotocols $\prod_S^{(x)}, \prod_S^{(y)}$ in (**a**), (**b**).

x-preserving (resp. y-preserving) strong t-privacy, with sharing parameter ℓ and share ownership functions $\mathcal{O}_x, \mathcal{O}_y$ and $\mathcal{O}_z^{(x)}$ (resp. $\mathcal{O}_z^{(y)}$). The basic idea of protocol $\prod$ is to share the circuit inputs using an ℓ-of-ℓ sharing, and then apply the shared 2-product subprotocol at each Mult gate of C, to arrive finally at a sharing of the circuit output value, which is made public.

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 $s_y(1) \cdots s_y(\ell) = 1_G$ of the identity element of G, 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 x-input to $\prod_S^{(y)}$), and the shared 1_G value (as the y-input to $\prod_S^{(y)}$). At the end of the subprotocol run, for each $j = 1, \ldots, \ell$, party $P_{\mathcal{O}_z^{(y)}(j)}$ holds an output share $s_z(j)$ and sends it to party $P_{\mathcal{O}_y(j)}$, resulting in an $\mathcal{O}_y$ sharing of x.

We now present the formal specification of the protocol $\prod$ as Algorithm 1. In this specification, we use the following terminology. For each node N of C, we call N an *x-node* (resp. *y-node*) if all outgoing edges of N are x-input edges (resp. y-input edges) of subsequent Mult nodes (by convention, we also define the output node of C as an x-input node). Otherwise, we call N a *mixed* node (for simplicity, we also define N as a mixed node if N has an outgoing edge which is an incoming edge to a CMult gate).

An example of the protocol is shown in Fig. 1.

Remark 4. As illustrated in the example on 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 [4,7], 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 $\prod(C, \prod_S^{(x)}, \prod_S^{(y)})$, assuming the correctness and strong t-privacy of subprotocols $\prod_S^{(x)}$,$\prod_S^{(y)}$. The idea is that due to the ℓ-of-ℓ sharing used for the inputs of C, the values of all but one share of each input of C can be simulated by independent random elements, and then the internal adversary view and all but one output share of each shared 2-product subprotocol run can be simulated using the subprotocol simulator. The detailed proof is in Appendix A.

Lemma 1. *For any m-input G-circuit C computing a function $f_C : G^m \to G$, 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 Definitions 2 and 3), then the protocol $\prod(C, \prod_S^{(x)}, \prod_S^{(y)})$ is an n-party t-private protocol for computing the function f_C.*

Algorithm 1 G-Circuit Protocol $\prod(C, \prod_S^{(x)}, \prod_S^{(y)})$

Input: For $i = 1, \ldots, m$, party $P_{j(i)}$ (for some $j(i) \in [n]$) holds input $x_i \in G$ at input node N_i of C.

1. For $i = 1, \ldots, m$, party $P_{j(i)}$ 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 node N_i of C is an x-node (resp. y-node), then for $k = 1, \ldots, \ell$, $P_{j(i)}$ sends the kth share $s_{x_i}(k)$ to party $P_{\mathcal{O}_x(k)}$ (resp. $P_{\mathcal{O}_y(k)}$), and we label the outgoing edges of N_i with $\boldsymbol{s}_{x_i}$. Otherwise, if N_i is a mixed node, then $P_{j(i)}$ generates another uniformly random ℓ-of-ℓ sharing $\boldsymbol{s}'_{x_i} = (s'_{x_i}(1), \ldots, s'_{x_i}(\ell))$ of $x_i = s'_{x_i}(1) \cdots s'_{x_i}(\ell)$, and sends for $k = 1, \ldots, \ell$, $s_{x_i}(k)$ to party $P_{\mathcal{O}_x(k)}$ and $s'_{x_i}(k)$ to party $P_{\mathcal{O}_y(k)}$, labeling the x-input outgoing edges of N_i with $\boldsymbol{s}_{x_i}$ and the y-input outgoing edges of N_i with $\boldsymbol{s}'_{x_i}$.

2. For each internal node N of C:

- If node N is an x-node (resp. y-node) Mult gate with x-input edge labeled by sharing $x = s_x(1) \cdots s_x(\ell)$ and y-input edge labeled by sharing $y = s_y(1) \cdots s_y(\ell)$, run 2-product subprotocol $\prod_S^{(c)}$ with $c = x$ (resp. $c = y$) on the sharings $(s_x(1), \ldots, s_x(\ell))$ and $(s_y(1), \ldots, s_y(\ell))$, resulting in an $\mathcal{O}_z^{(c)}$-sharing of node N's output value $x \cdot y$. Then, for each $j = 1 \ldots, \ell$, party $P_{\mathcal{O}_z^{(c)}(j)}$ sends its output share to party $P_{\mathcal{O}_x(j)}$ (resp. party $P_{\mathcal{O}_y(j)}$), resulting in an $\mathcal{O}_x$ (resp. $\mathcal{O}_y$) sharing of the output value. We label the outgoing x-input (resp. y-input) edges of N with this sharing. Otherwise, if N is a mixed node Mult gate, run subprotocol $\prod_S^{(x)}$ to get an $\mathcal{O}_x$-sharing of output value $x \cdot y$ as in the case that N is an x-node, and label all outgoing edges except the y-input edges with this sharing. Then, run subprotocol Convert to obtain an $\mathcal{O}_y$-sharing of output value $x \cdot y$, and label the y-input outgoing edges with this sharing.
- 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 $s_x(\ell) \cdot \beta$.

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

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

Output: Every party knows the function output value $f_C(x_1, \ldots, x_m)$.

2.4. *Reducing Shared 2-Product Protocols to Graph Coloring*

2.4.1. *Graph Coloring Problem*

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 the PDAG $\mathcal{G}$ to represent a black-box protocol, where the input/output nodes are labeled by the protocol input/output group elements, and the internal graph nodes are labeled by intermediate protocol values. Each internal graph node is also assigned a *color* 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 (from the leftmost incoming edge to the rightmost one) and resharing the product along the outgoing edges using a k-of-k secret-sharing scheme (see scheme in Proposition 1 of Sect. 2.1.2). All computations in the ith round of the 2-product subprotocol correspond to the ith row (from the top) in the PDAG. Communication between nodes corresponds 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 (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 $1, \ldots, m$. 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 $\eta_1^{(i,j)} < \cdots < \eta_{q^{(i,j)}}^{(i,j)}$ 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-coloring function that associates to each node (i, j) of $\mathcal{G}$ a color $C(i, j)$ chosen from a set of n possible colors $[n]$. We now define the notion of a t-Reliable n-Coloring.

Definition 5 (t-Reliable n-Coloring). We say that $C : [m] \times [m] \to [n]$ is a *t-Reliable n-Coloring* for the admissible PDAG $\mathcal{G}$ (with share parameter ℓ and size parameter m) if for each t-color subset $I \subset [n]$, there exist $j_x^*, j_y^*, j_z^* \in [\ell]$ with $j_z^* \in \{j_x^*, j_y^*\}$ such that we have the following.

- 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 colors 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.

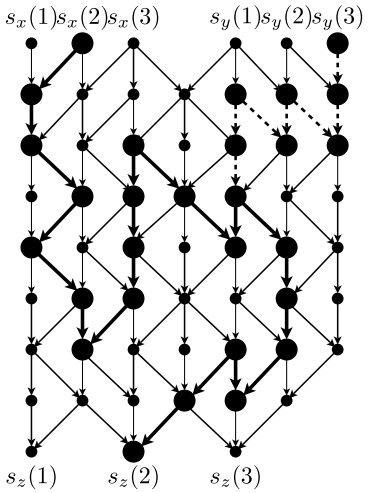


Fig. 2. An admissible PDAG with sharing parameter $\ell = 3$ (node colors are not indicated). For a given collusion I, an example I-avoiding path $PATH_x$ is shown in *heavy black*, and an example I-avoiding path $PATH_y$ (until the meeting with $PATH_x$) is shown in *dashed black*. Here, we have $j_x^* = j_z^* = 2$ and $j_y^* = 3$.

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-Coloring. If $j_z^* = j_x^* = j_y^*$ for all I, then we say that C is a *Symmetric* t-Reliable n-Coloring.

Remark 5. The paths $PATH_x$ and $PATH_y$ in Definition 5 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.

The communication of our MPC protocols will be represented by the number of edges of the admissible PDAG (see Algorithm 2). Based on the following result by Euler, we only need to focus our interest on the number of vertices of the graph.

Theorem 2 [14]. *Let $\mathcal{G}$ be a planar graph on V vertices. If $V \geq 3$, then the number of edges of $\mathcal{G}$ is at most $3V - 6$.*

2.4.2. *Construction of a t-Private n-Party Shared 2-Product Subprotocol from a t-Reliable n-Coloring of a Planar Graph*

We are to reduce the problem of constructing a t-private n-party shared 2-product subprotocol $\prod_S$ to the combinatorial problem of finding a t-reliable n-coloring of the nodes of a planar graph. We note that our notion of a 't-reliable n-coloring' is closely related to a similar notion defined in [12], 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 colors and the adversary controls all nodes with colors belonging to a t-color subset I.

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

Input: An admissible PDAG $\mathcal{G}$ with share parameter ℓ and size parameter m, an associated t-reliable n-coloring C and two values x and y.

1. We define the share ownership functions $\mathcal{O}_x, \mathcal{O}_y, \mathcal{O}_z$ of $\prod_S(\mathcal{G}, C)$ according to the colors 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.

2. 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 $\eta_1^{(i,j)} < \cdots < \eta_{q^{(i,j)}}^{(i,j)}$ 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 an $q^{(i,j)}$-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 $s_k^{(i,j)}$.

Output: ℓ shares of the product $z = x \cdot y$.

Denote by $\mathcal{G}$ a PDAG 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 $\mathcal{G}$ to represent a black-box protocol, where the input/output nodes are labeled by the protocol input/output group elements, and the internal graph nodes are labeled by intermediate protocol values. Each internal graph node is also assigned a *color* 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 the multiplication of the values on all incoming edges followed by a re-sharing of the product along the outgoing edges using the k-of-k secret-sharing scheme in Proposition 1. All computations in the ith round of the 2-product subprotocol cor-

respond to the ith row (from the top) in the PDAG. Communications between nodes correspond to edges between consecutive rows.

Given an admissible PDAG $\mathcal{G}$ (with share parameter ℓ and size parameter m) and an associated t-reliable n-coloring $C : [m] \times [m] \to [n]$, we construct a t-private n-party shared 2-product subprotocol $\prod_S(\mathcal{G}, C)$ as represented in Algorithm 2.

Note that the correctness of $\prod_S$ 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. The full proof of the security claim of Lemma 2 can be found in Appendix B. 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. 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 $\mathsf{S}_{\prod_S}$, we may assume, by Proposition 1, that in the real subprotocol $\prod_S$, 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 $\mathsf{S}_{\prod_S}$ 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 $\mathsf{S}_{\prod_S}$ 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 $\mathsf{S}_{\prod_S}$, 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 $\mathsf{S}_{\prod_S}$. Since $\mathsf{S}_{\prod_S}$ knows all inputs to $\prod_S$ it can compute all other edge values in the $\prod_S$, including all outputs except the j_z^*th one (which is on $PATH_x$ and $PATH_y$), as required.

Lemma 2. *If $\mathcal{G}$ is an admissible PDAG and C is an x-preserving (resp. y-preserving) t-reliable n-coloring for $\mathcal{G}$ then $\prod_S(\mathcal{G}, C)$ achieves x-preserving (resp. y-preserving) strong t-privacy. Moreover, if C is a* symmetric *t-reliable n-coloring, then $\prod_S(\mathcal{G}, C)$ achieves* symmetric *strong t-privacy.*

3. Graph Coloring Problem and Constructions

3.1. *A Family of PDAGs*

In our coloring constructions, we focus on a particularly simple admissible PDAG $\mathcal{G}_{\text{tri}}(\ell', \ell)$ defined as follows. This graph has sharing parameter ℓ and has $\ell' \times \ell$ nodes. It is shown in Fig. 3. The nodes of $\mathcal{G}_{\text{tri}}(\ell', \ell)$ 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 $\mathcal{G}_{\text{tri}}(\ell', \ell)$:

- [horizontal edges] for $i \in [\ell']$ and for $j \in [\ell - 1]$, there is a directed edge from node $(i, j+1)$ to (i, j),
- [vertical edges] for $i \in [\ell' - 1]$ and for $j \in [\ell]$, there is a directed edge from node (i, j) to node $(i+1, j)$,
- [diagonal edges] for $i \in [\ell' - 1]$ and for $j \in \{2, \ldots, \ell\}$, there is a directed edge from node (i, j) to node $(i+1, j-1)$.

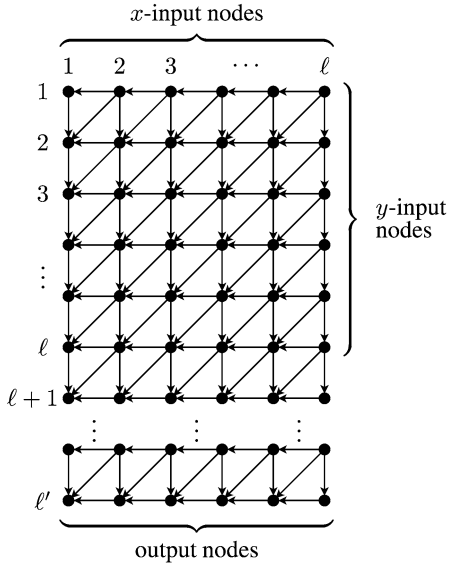


Fig. 3. The PDAG $\mathcal{G}_{\text{tri}}(\ell', \ell)$.

Remark 6. The ℓ nodes on the top row (row 1) of $\mathcal{G}_{\text{tri}}$ are the x-input nodes, indexed from left to right. The top ℓ nodes on the rightmost column of $\mathcal{G}_{\text{tri}}$ (column ℓ) are the y-input nodes, indexed from top to bottom.

Remark 7. The reader may notice that the above specification of $\mathcal{G}_{\text{tri}}$ does not formally satisfy the convention for drawing an admissible PDAG as exposed in Definition 4, 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 $\mathcal{G}_{\text{tri}}$ can also be drawn strictly according to Definition 4. Namely, by rotating the drawing of $\mathcal{G}_{\text{tri}}$ in Fig. 3 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 color as the corresponding input nodes of $\mathcal{G}_{\text{tri}}$. 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. 3 for clarity.

Remark 8. All diagonal edges in the definition of $\mathcal{G}_{\text{tri}}$ above are parallel (with a 'positive slope', when using the drawing convention in Fig. 3). However, it is clear that the admissible PDAG requirements are still satisfied if we remove from $\mathcal{G}_{\text{tri}}$ 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 'generalized' PDAGs by $\mathcal{G}_{\text{gtri}}$.

3.2. *Relaxing the Coloring Conditions*

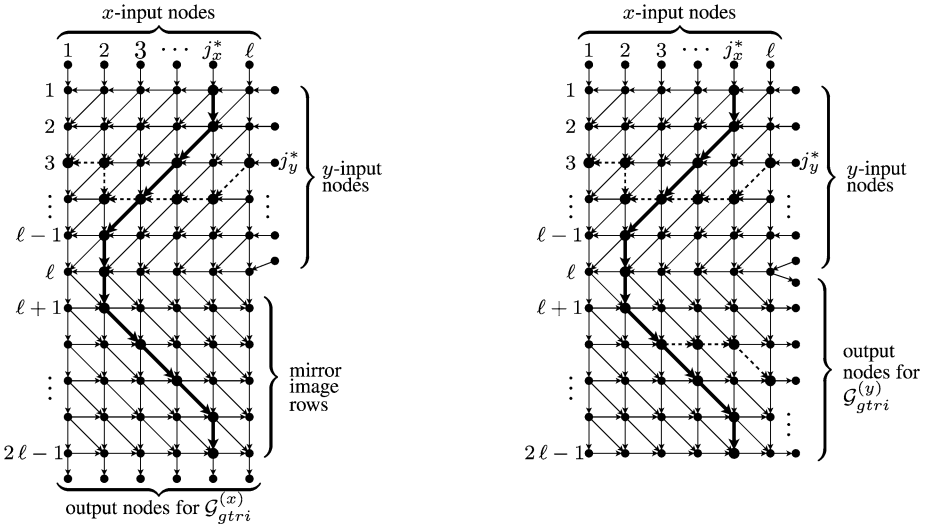
As a first step toward our constructions, we relax the properties required from the coloring in Definition 5 to slightly simpler requirements for the square graph $\mathcal{G}_{\text{tri}}(\ell, \ell)$ (i.e. $\ell' = \ell$), as follows.

Definition 6 (Weakly t-Reliable n-Coloring). We say that $C : [\ell] \times [\ell] \to [n]$ is a *Weakly* t-Reliable n-Coloring for graph $\mathcal{G}_{\text{tri}}(\ell, \ell)$ if for each t-color 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.

Remark 9. 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-coloring for the square graph $\mathcal{G}_{\text{tri}}(\ell, \ell)$ is sufficient for constructing a (standard) t-reliable n-coloring for a rectangular graph $\mathcal{G}_{\text{gtri}}(2\ell - 1, \ell)$. The idea is to add $\ell - 1$ additional rows to $\mathcal{G}_{\text{tri}}(\ell, \ell)$ by appending a 'mirror image' (reflected about the last row) of itself, as shown in Fig. 4. Note



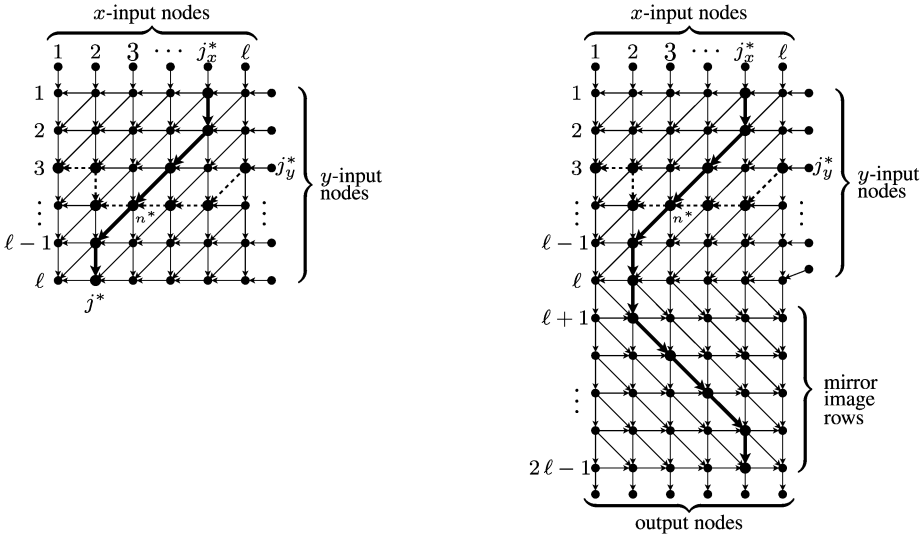
(a) $\mathcal{G}_{gtri}^{(x)}(2\ell - 1, \ell)$ (b) $\mathcal{G}_{gtri}^{(y)}(2\ell - 1, \ell)$

Fig. 4. Illustration of the two Versions of $\mathcal{G}_{\text{gtri}}(2\ell - 1, \ell)$.

that we define two versions of $\mathcal{G}_{\text{gtri}}(2\ell-1,\ell)$ called $\mathcal{G}^{(x)}_{\text{gtri}}(2\ell-1,\ell)$ and $\mathcal{G}^{(y)}_{\text{gtri}}(2\ell-1,\ell)$. The difference between them is only in the choice of output nodes: the bottom row ℓ nodes are used as output nodes for $\mathcal{G}^{(x)}_{\text{gtri}}(2\ell-1,\ell)$ as shown in Fig. 4(a) (giving an x-preserving coloring) and the last ℓ nodes on the rightmost column are used as output nodes for $\mathcal{G}^{(y)}_{\text{gtri}}(2\ell-1,\ell)$ as shown in Fig. 4(b) (giving an y-preserving coloring). Note that the color of the input and output nodes is identical to the color of the nodes connected to them.

Lemma 3. *Let $C:[\ell]\times[\ell]\to[n]$ be a weakly t-reliable n-coloring (see Definition 6) for square admissible PDAG $\mathcal{G}_{\text{tri}}(\ell,\ell)$. Then, we can construct an x-preserving (resp. y-preserving) t-reliable n-coloring (see Definition 5) for a rectangular admissible PDAG $\mathcal{G}^{(x)}_{\text{gtri}}(2\ell-1,\ell)$ (resp. $\mathcal{G}^{(y)}_{\text{gtri}}(2\ell-1,\ell)$). Moreover, the 2-product subprotocols $\prod^{(x)}_S$, $\prod^{(y)}_S$ constructed from $\mathcal{G}^{(x)}_{\text{gtri}}(2\ell-1,\ell)$ and $\mathcal{G}^{(y)}_{\text{gtri}}(2\ell-1,\ell)$ (using Lemma 2) are compatible (see Definition 3).*

Proof. For each t-color subset I, let P_x and P_y denote the top-bottom and right–left I-avoiding paths in $\mathcal{G}_{\text{tri}}(\ell,\ell)$ for the given coloring 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. 5(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 $\mathcal{G}_{\text{tri}}(\ell,\ell)$



(a) $\mathcal{G}_{tri}(\ell,\ell)$ (b) $\mathcal{G}^{(y)}_{gtri}(2\ell-1,\ell)$

Fig. 5. Example of paths in a square PDAG with parameter ℓ for a given weakly t-reliable n-coloring (P_x (*heavy black*) and P_y (*dashed black*) intersect at node n^*) and in its corresponding rectangular grid.

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 $\mathcal{G}_{\text{gtri}}^{(x)}(2\ell-1,\ell)$ as follows. The first ℓ rows of $\mathcal{G}_{\text{gtri}}^{(x)}(2\ell-1,\ell)$ and interconnecting edges are identical to $\mathcal{G}_{\text{tri}}(\ell,\ell)$. For the bottom $\ell-1$ rows of $\mathcal{G}_{\text{gtri}}^{(x)}(2\ell-1,\ell)$, we take the 'mirror image' reflection of the top ℓ rows and their interconnecting edges, reflected about the ℓth row. The outputs are taken from the bottom ℓ nodes (see Fig. 5(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-coloring $C' : [2\ell-1]\times[\ell] \to [n]$ of $\mathcal{G}_{\text{gtri}}^{(x)}(2\ell-1,\ell)$ similarly, i.e. the top ℓ rows of $\mathcal{G}_{\text{gtri}}^{(x)}(2\ell-1,\ell)$ are colored using C, i.e. $C'(i,j) = C(i,j)$ for $(i,j) \in [\ell]\times[\ell]$, and the last $\ell-1$ rows of $\mathcal{G}_{\text{gtri}}^{(x)}(2\ell-1,\ell)$ are colored 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-coloring for $\mathcal{G}_{\text{gtri}}^{(x)}(2\ell-1,\ell)$. Indeed, thanks to the 'mirror symmetry' of C' and $\mathcal{G}_{\text{gtri}}^{(x)}(2\ell-1,\ell)$ 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 $\mathcal{G}_{\text{gtri}}^{(x)}(2\ell-1,\ell)$. 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. 4(b). Thus, C' satisfies the requirements of an x-preserving t-reliable n-coloring for $\mathcal{G}_{\text{gtri}}^{(x)}(2\ell-1,\ell)$.

The PDAG $\mathcal{G}_{\text{gtri}}^{(y)}(2\ell-1,\ell)$ is identical to $\mathcal{G}_{\text{gtri}}^{(y)}(2\ell-1,\ell)$ except that the outputs are taken from the ℓ bottom nodes of the last column as shown in Fig. 5(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_y^*th node, i.e. output node $(2\ell - j_y^*, \ell)$ of $\mathcal{G}_{\text{gtri}}^{(y)}(2\ell-1,\ell)$) that C' is a y-preserving t-reliable n-coloring for $\mathcal{G}_{\text{gtri}}^{(y)}(2\ell-1,\ell)$. Moreover, since for all collusions $PATH_x$ and $PATH_y$ are identical in the top half of $\mathcal{G}_{\text{gtri}}^{(y)}(2\ell-1,\ell)$ and $\mathcal{G}_{\text{gtri}}^{(x)}(2\ell-1,\ell)$, we see that j_x^* and j_y^* are the same in the two graphs so the corresponding subprotocols $\Pi_S^{(x)}$ and $\Pi_S^{(y)}$ are compatible, as claimed. □

3.3. *Deterministic Constructions*

We now present two constructions of t-reliable n-colorings 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 Sects. 2.3 and 2.4.2. Our first deterministic construction achieves optimal collusion security ($t < n/2$) but has exponential complexity ($\ell = \binom{n}{t}$). However, when used recursively, it will lead to a deterministic scheme with polynomial communication complexity for any $t \in O(n^{1-\epsilon})$ where ϵ is any positive constant.

First Construction C_{comb} ($t < n/2$ and $\ell = \binom{n}{t}$) We now present an explicit construction of a t-reliable n-coloring C_{comb} of the square graph $\mathcal{G}_{\text{tri}}(\ell,\ell)$. The construction

applies for all $n \geq 2t+1$ (i.e. $t \leq \lfloor \frac{n-1}{2} \rfloor$), and hence the n-product protocol constructed from it by the method of Sects. 2.3 and 2.4.2 achieves $\lfloor \frac{n-1}{2} \rfloor$-privacy (which is optimal since honest majority is needed as argued in Sect. 2.2).

Lemma 4. *For $n \geq 2t+1$, the coloring C_{comb} is a symmetric t-reliable n-coloring for the graph $\mathcal{G}_{\text{tri}}(\ell, \ell)$, with $\ell = \binom{n}{t}$.*

Proof. Given each t-color subset $I \subseteq [n]$, let j^* denote the index of I in the sequence $I_1, \ldots, I_\ell$ of all t-color subsets used to construct C_{comb}, i.e $I_{j^*} = I$. By construction of C_{comb}, none of the nodes of $\mathcal{G}_{\text{tri}}(\ell, \ell)$ along column j^* have colors in $I_{j^*} = I$. Hence one can take column j^* of $\mathcal{G}_{\text{tri}}(\ell, \ell)$ as $PATH_x$. Similarly, we also know that none of the nodes of $\mathcal{G}_{\text{tri}}(\ell, \ell)$ along row j^* have colors 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-coloring for graph $\mathcal{G}_{\text{tri}}(\ell, \ell)$, as required. □

Remark 10. The coloring C_{comb} remains a symmetric t-reliable n-coloring even if we remove all diagonal edges from $\mathcal{G}_{\text{tri}}(\ell, \ell)$ (since the paths $PATH_x$ and $PATH_y$ only contain vertical and horizontal edges).

Combining Lemma 4 (applied to a subset of $n' = 2t+1 \leq n$ colors from $[n]$) with Lemma 1 in Sect. 2.3 and Lemma 2 in Sect. 2.4.2, we have the following.

Corollary 1. *For any $t < n/2$, there exists a black-box t-private protocol for G-circuits with communication complexity $O(\binom{2t+1}{t}^2 N_g)$ group elements. This protocol runs in $4\binom{2t+1}{t}(N_g - 1)$ rounds.*

Proof. It remains to argue about the round complexity. The admissible PDAG used for this protocol is $\mathcal{G}_{\text{gtri}}(2\ell - 1, \ell)$ where $\ell = \binom{2t+1}{t}$. By construction, the top ℓ rows of $\mathcal{G}_{\text{gtri}}(2\ell - 1, \ell)$ represent $\mathcal{G}_{\text{tri}}(\ell, \ell)$. As said in the second paragraph of Sect. 2.4.2, the set of directed arrows outgoing a given node $\mathcal{N}$ represents a secret-sharing scheme whose dealer is the player corresponding to the color of $\mathcal{N}$. Thus, no node/player can perform such a sharing before receiving information from its all parent nodes represented by the incoming edges of $\mathcal{N}$. In the case of $\mathcal{G}_{\text{tri}}(\ell, \ell)$, every node has at most three parents and the only node without parents is the top-right node.

Let us number the rows of $\mathcal{G}_{\text{tri}}(\ell, \ell)$ from top to bottom and its columns from left to right. The player corresponding to the node at row 1 and column ℓ is to initiate the sharing process. This phenomenon is to propagate throughout $\mathcal{G}_{\text{tri}}(\ell, \ell)$. After one round of communication, nodes $(1, \ell - 1)$ and $(2, \ell)$ have received information from all their parents. After two rounds, nodes $(1, \ell - 2)$, $(2, \ell - 1)$ and $(3, \ell)$ are ready to share in turn. One can see that using 2ℓ rounds results in covering all of $\mathcal{G}_{\text{tri}}(\ell, \ell)$.

The previous propagation phenomenon moves diagonally from right to left. When we consider the bottom ℓ rows of $\mathcal{G}_{\text{gtri}}(2\ell - 1, \ell)$, they constitute a mirror of the upper part $\mathcal{G}_{\text{tri}}(\ell, \ell)$. Thus, another propagation phenomenon occurs from left to right. Thus, 4ℓ rounds are enough perform the computation represented by an instance of $\mathcal{G}_{\text{gtri}}(2\ell -$

$1, \ell)$. This grid is to be used $(N_g - 1)$ times to perform the computation of the G-circuit which implies at total of $4\ell(N_g - 1)$ rounds. □

Remark 11. The previous round complexity analysis is unchanged if $\mathcal{G}_{\text{tri}}(\ell, \ell)$ has no diagonal edges.

Remark 12. One may also consider more general adversary structures in place of the t-threshold structure. This construction immediately generalizes 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 [20], and gives a coloring for the graph $\mathcal{G}_{\text{tri}}(\ell, \ell)$ with $\ell = |\mathcal{A}|$.

Second Construction ($t = O(n^{1-\epsilon})$ and $\ell = poly(n)$) We will use the previous scheme as a building block. As noticed in the Remark 10, even if we remove the diagonal edges from $\mathcal{G}_{\text{tri}}(\ell, \ell)$, we still had the existence of I-avoiding top-bottom and right–left paths. Thus, we can assume that $\mathcal{G}_{\text{rec}}(\ell)$ has no such edges so that $\mathcal{G}_{\text{rec}}(\ell)$ is a square grid the side length of which is ℓ nodes. $\mathcal{G}_{\text{rec}}(\ell)$ is an admissible PDAG.

First, we will focus on particular pairs (t, n). Second, we generalize our result to any (t, n) with $t = O(n^{1-\epsilon})$.

We recursively construct our admissible PDAG $\mathcal{G}_{\text{rec}}$ and its coloring C_{rec}. Let $d \in \mathbb{N} \setminus \{0, 1\}$ be a constant. Denote by $\mathcal{B}_d$ the binomial coefficient $\binom{2d-1}{d-1}$.

Theorem 3. *For any positive integer k, there is a weakly t_k-reliable n_k-coloring $C_{\text{rec}}(\ell_k)$ for the square admissible* PDAG $\mathcal{G}_{\text{rec}}(\ell_k)$ *with e_k edges and v_k vertices, where the parameters are*:

$$\begin{cases} t_k = d^k - 1, \\ n_k = (2d-1)^k, \\ \ell_k = \mathcal{B}_d^k(\mathcal{B}_d + 1)^{k-1}, \\ e_k = 2(\mathcal{B}_d - 1)\mathcal{B}_d^{2k-1} + \frac{8\mathcal{B}_d^3 - 4\mathcal{B}_d^2 - \mathcal{B}_d}{4}\mathcal{B}_d^{2k-3}\sum_{i=1}^{k-1}(4\mathcal{B}_d)^i, \\ v_k = 2^{2k-2}\mathcal{B}_d^{3k-1}. \end{cases}$$

Proof. We prove the theorem by induction on k.

$k = 1$: We have $t_1 = d - 1$, $n_1 = 2d - 1$ and $\ell_1 = \mathcal{B}_d$. We set $\mathcal{G}_{\text{rec}}(\ell_1) := \mathcal{G}_{\text{tri}}(\ell_1, \ell_1)$. We define $C_{\text{rec}}(\ell_1)$ as being the combinatorial coloring C_{comb} (see Algorithm 3). Since there are no diagonal edges in $\mathcal{G}_{\text{tri}}(\ell_1, \ell_1)$, we have $e_1 = 2(\ell_1 - 1)\ell_1 = 2(\mathcal{B}_d - 1)\mathcal{B}_d$ and $v_1 = \ell_1^2 = \mathcal{B}_d^2$.

$k \geq 1$: Suppose we already have the construction and coloring for k, we recursively construct $\mathcal{G}_{\text{rec}}(\ell_{k+1})$ from $\mathcal{G}_{\text{rec}}(\ell_k)$.

We first build the block grid B by copying $4\mathcal{B}_d$ times $\mathcal{G}_{\text{rec}}(\ell_1)$ and representing those copies as the external rows and columns of an $(\mathcal{B}_d + 1) \times (\mathcal{B}_d + 1)$ grid. Note that, on this 'large' grid, each node is in fact a copy of $\mathcal{G}_{\text{rec}}(\ell_1)$. The connections between two copies of $\mathcal{G}_{\text{rec}}(\ell_1)$ are as follows. Horizontally (top and bottom rows), we draw a directed edge from node $(i, 1)$ in the right-hand side copy to node (i, ℓ_1) in the left-hand side copy for $i \in [\ell_1]$ (i.e. we horizontally connect nodes at the same level). Vertically

Algorithm 3 Coloring C_{comb}

Input: An $\ell \times \ell$ grid where $\ell = \binom{n}{t}$ and $n = 2t + 1$.

1. Let $I_1, \ldots, I_\ell$ denote the sequence of all t-color subsets of $[n]$ (in some ordering).
2. For each $(i, j) \in [\ell] \times [\ell]$, define the color $C(i, j)$ of node (i, j) in the grid to be any color in the set $S_{i,j} := [n] \setminus (I_i \cup I_j)$.

Output: A n-coloring of the grid.

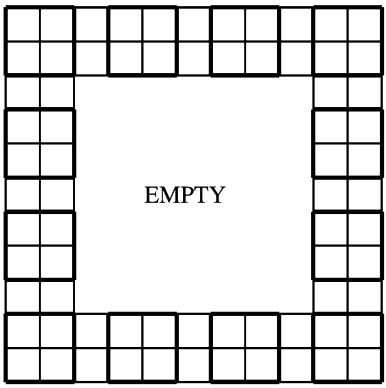


Fig. 6. Block B when $d = 2$.

(left and right columns), we draw a directed edge from node (ℓ_1, j) in the top-side copy to node $(1, j)$ in the bottom side copy for $j \in [\ell_1]$ (i.e. we vertically connect nodes at the same level). This is illustrated as Fig. 6 in the case $d = 2$.

The block B is an $(\mathcal{B}_d(\mathcal{B}_d + 1)) \times (\mathcal{B}_d(\mathcal{B}_d + 1))$ grid with $4\mathcal{B}_d v_1$ vertices. It exhibits the following property.

Proposition 2. *The block grid B admits an $(2d - 1)$-coloring (just use the same C_{comb} for each copy of $\mathcal{G}_{\text{rec}}(\ell_1)$), such that for any $(d - 1)$-color subset $I \subset [2d - 1]$, there are $\mathcal{B}_d + 1$ horizontal (vertical) I-avoiding path in B.*

Remark 13. We need to explain the meaning of *horizontal* and *vertical* I-avoiding paths in B. This block is represented over an $(\mathcal{B}_d(\mathcal{B}_d + 1)) \times (\mathcal{B}_d(\mathcal{B}_d + 1))$ grid of nodes. Thus, we can number every node of B by a pair $(i, j) \in [\mathcal{B}_d(\mathcal{B}_d + 1)]^2$. Note that not all these values are used since B only has $4\mathcal{B}_d^3$ nodes (EMPTY space on Fig. 6). A horizontal I-avoiding path is a path starting at a node $(i, 1)$ (leftmost column of block B) and finishing at node $(i, \mathcal{B}_d(\mathcal{B}_d + 1))$ (rightmost column of block B). The reader may notice that such a path is not unique. However, we abusively identify the (non-unique) path to the pair of nodes $\{(i, 1), (i, \mathcal{B}_d(\mathcal{B}_d + 1))\}$. Thus, the previous proposition should rather be stated as the number of pairs $\{(i, 1), (i, \mathcal{B}_d(\mathcal{B}_d + 1))\}$ (respectively, $\{(1, j), (\mathcal{B}_d(\mathcal{B}_d + 1), j)\}$) being equal to $\mathcal{B}_d + 1$. We used this abuse in order to simplify the following demonstrations.

After this vocabulary explanation, the demonstration of Proposition 2 becomes straightforward.

Proof. As said in Remark 10, we considered for our recursive construction that $\mathcal{G}_{\text{tri}}(\ell, \ell)$ had no diagonal edges. Let I be a $(d-1)$-color subset of $[2d-1]$. By Lemma 4, there exist a I-avoiding top-bottom path and a I-avoiding right–left path in $\mathcal{G}_{\text{rec}}(\ell_1)$ which are straight lines.

Since B is a-$(\mathcal{B}_d+1)\times(\mathcal{B}_d+1)$-copy of $\mathcal{G}_{\text{rec}}(\ell_1)$ and, due to the vertical/horizontal connections of these copies, we deduce that there are $(\mathcal{B}_d+1)I$-avoiding top-bottom paths and $(\mathcal{B}_d+1)I$-avoiding right–left paths in B. Moreover, each of these paths is a horizontal/vertical line with the abuse of terminology explained in the previous remark. (End of proof of Proposition 2.) □

Now, we construct $\mathcal{G}_{\text{rec}}(\ell_{k+1})$ and its coloring $C_{\text{rec}}(\ell_{k+1})$ as follows. We replace each node in $\mathcal{G}_{\text{rec}}(\ell_k)$ by a copy of B. If the node of $\mathcal{G}_{\text{rec}}(\ell_k)$ was colored by the color $c\in[n_k]$, then we color B with the set of colors $\{(2d-1)(c-1)+1, (2d-1)(c-1)+2,\ldots,(2d-1)c\}$, using C_{comb}. All the edges within each copy of B remain identical in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$. We deduce from this constructing technique that the number of edges v_{k+1} of $\mathcal{G}_{\text{rec}}(\ell_{k+1})$ is equal to $(4\mathcal{B}_d v_1)v_k$. In other words: $v_{k+1}=4\mathcal{B}_d v_1 2^{2k-2}\mathcal{B}_d^{3k-1}=2^{2k}\mathcal{B}_d^{3k+2}$.

Now, we show how to connect two copies of B. We first focus on vertical connections. Consider an edge in $\mathcal{G}_{\text{rec}}(\ell_k)$ from a node in the ith row to another node in the $(i+1)$th row. Since these two nodes have been replaced by two copies of B, we denote the nodes on the top copy (i.e. those corresponding to the nodes of the ith row in $\mathcal{G}_{\text{rec}}(\ell_k)$) by $v_{1,1},\ldots,v_{1,\mathcal{B}_d}, v_{2,1},\ldots,v_{\mathcal{B}_d+1,\mathcal{B}_d}$ and the nodes on the bottom copy as $w_{1,1},\ldots,w_{1,\mathcal{B}_d}, w_{2,1},\ldots,w_{\mathcal{B}_d+1,\mathcal{B}_d}$.

For each $(i,j)\in[\mathcal{B}_d]\times[\mathcal{B}_d]$, we add a directed edge $(v_{i,j}, w_{i,j+i-1})$ in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$. If the index $(j+i-1)$ is greater than $\mathcal{B}_d$, $w_{i,j+i-1}$ is the node $w_{i+1,j+i-1-\mathcal{B}_d}$. Figure 7 gives the example for $d=2$. The connection process works similarly for two consecutive columns where we replace each horizontal edge from $\mathcal{G}_{\text{rec}}(\ell_k)$ by $\mathcal{B}_d^2$ different edges in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$.

It is clear that the number of nodes on each side of the square $\mathcal{G}_{\text{rec}}(\ell_{k+1})$ is:

$$\ell_{k+1}=\mathcal{B}_d(\mathcal{B}_d+1)\ell_k=\mathcal{B}_d^{k+1}(\mathcal{B}_d+1)^k$$

and the number of colors used in $C_{\text{rec}}(\ell_{k+1})$ is $n_{k+1}=(2d-1)n_k=(2d-1)^{k+1}$. The grid $\mathcal{G}_{\text{rec}}(\ell_{k+1})$ obtained by this recursive process is also an admissible PDAG due to the horizontal/vertical connection processes between two copies of B (as well as two copies of $\mathcal{G}_{\text{rec}}(\ell_1)$ inside B).

Let us focus on the value of e_{k+1}. Each edge of $\mathcal{G}_{\text{rec}}(\ell_k)$ is turned into $\mathcal{B}_d^2$ edges in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$ and each node in $\mathcal{G}_{\text{rec}}(\ell_k)$ becomes of block B in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$.

A way to look at B is to consider a cycle of $4\mathcal{B}_d$ nodes (each of them being a copy of $\mathcal{G}_{\text{rec}}(\ell_1)$) where every consecutive pair of nodes is linked $\mathcal{B}_d$ times. Based on this

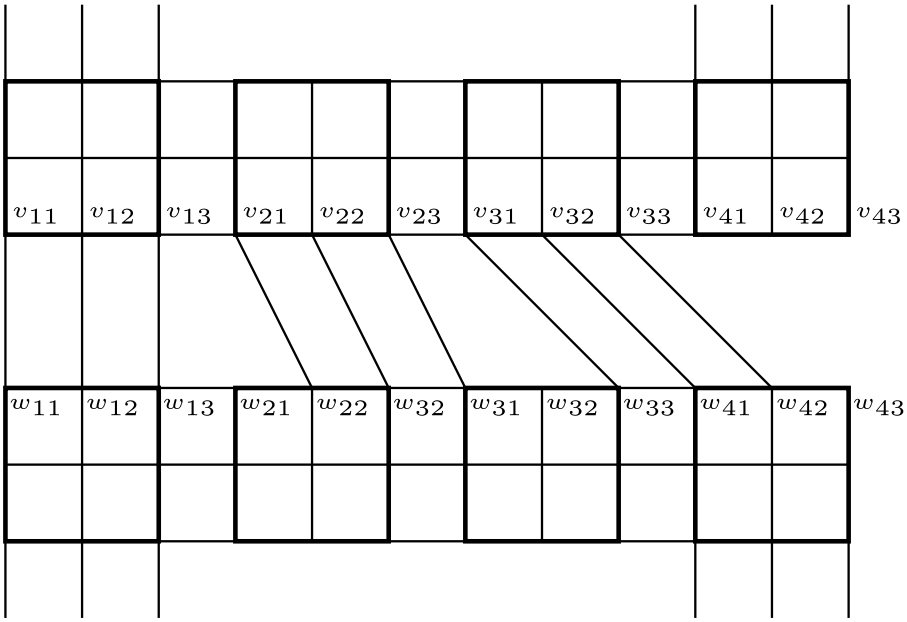


Fig. 7. How to vertically connect two copies of B when $d = 2$.

observation, we deduce that each block B contains $(8\mathcal{B}_d^3 - 4\mathcal{B}_d^2 - \mathcal{B}_d)$ edges. We get

$$\begin{aligned} e_{k+1} &= \mathcal{B}_d^2 e_k + \left(8\mathcal{B}_d^3 - 4\mathcal{B}_d^2 - \mathcal{B}_d\right) v_k \\ &= 2(\mathcal{B}_d - 1)\mathcal{B}_d^{2k+1} + \frac{8\mathcal{B}_d^3 - 4\mathcal{B}_d^2 - \mathcal{B}_d}{4}\mathcal{B}_d^{2k-1} \\ &\quad \times \sum_{i=1}^{k-1} (4\mathcal{B}_d)^i + \left(8\mathcal{B}_d^3 - 4\mathcal{B}_d^2 - \mathcal{B}_d\right)\frac{2^{2k}}{4}\mathcal{B}_d^{3k-1} \\ &= 2(\mathcal{B}_d - 1)\mathcal{B}_d^{2k+1} + \frac{8\mathcal{B}_d^3 - 4\mathcal{B}_d^2 - \mathcal{B}_d}{4}\mathcal{B}_d^{2k-1}\left[\sum_{i=1}^{k-1} (4\mathcal{B}_d)^i + 4^k\mathcal{B}_d^k\right] \\ &= 2(\mathcal{B}_d - 1)\mathcal{B}_d^{2k+1} + \frac{8\mathcal{B}_d^3 - 4\mathcal{B}_d^2 - \mathcal{B}_d}{4}\mathcal{B}_d^{2k-1}\sum_{i=1}^{k} (4\mathcal{B}_d)^i. \end{aligned}$$

The last point to prove is that for any t_{k+1}-color subset $I \subset [n_{k+1}]$, there is an I-avoiding top-bottom (and right–left) path in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$. We only prove the existence of a top-bottom path in this paper as the demonstration of the existence for a right–left path is similar. For each $j \in [n_k]$, we define the set I_j as:

$$I_j := I \cap \left\{(2d-1)(j-1)+1, (2d-1)(j-1)+2, \ldots, (2d-1)j\right\}.$$

Since

$$|I_1| + \cdots + |I_{n_k}| = |I| = t_{k+1} = d^{k+1} - 1 \tag{2}$$

and each $|I_j| \leq 2d - 1$, there are at least $(n_k - t_k)$ subsets having at most $(d - 1)$ elements. Indeed, in the opposite case, we would have

$$|I_1| + \cdots + |I_{n_k}| \geq d\big(n_k - (n_k - t_k - 1)\big) = d \times d^k = d^{k+1},$$

which would contradict equality (2). Assume that $S \subseteq [n_k]$ is the set of these indices (i.e. for each $j \in S$, $|I_j| \leq d - 1$). We have $|[n_k] \setminus S| \leq t_k$. By the induction hypothesis, there is an $([n_k] \setminus S)$-avoiding top-bottom path in $\mathcal{G}_{\text{rec}}(\ell_k)$, i.e., the colors used on this path all belong to S. Let $v_1, \ldots, v_m$ be the vertices of the path and denote the color of node v_j by $c_j \in S$ $(j \in [m])$.

Now, we show there is an I-avoiding top-bottom path in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$. In $\mathcal{G}_{\text{rec}}(\ell_{k+1})$, each node v_j has been replaced by a copy B_{v_j} with colors in $\{(2d-1)(c_j - 1) + 1, (2d-1)(c_j - 1) + 2, \ldots, (2d-1)c_j\}$. Since the color set I_{c_j} satisfies $|I_{c_j}| \leq d - 1$, by Proposition 2 we deduce that there are $\mathcal{B}_d$ horizontal and $\mathcal{B}_d$ vertical I_{c_j}-avoiding paths in B_{v_j}.

One can show that this property involves the existence of an I-avoiding top-bottom path in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$. This top-bottom path is the connection of an I_{c_1}-avoiding path (from B_{v_1}), an I_{c_2}-avoiding path (from B_{v_2}), …, an I_{c_m}-avoiding path (from B_{v_m}). The idea to connect these different paths is to focus on the respective positions of pairs of consecutive blocks. Roughly speaking, it works as follows. The I-avoiding top-bottom path in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$ originates from a node $\mathcal{N}_1$ located on the top side of B_{v_1}. Assume that B_{v_2} is located on the bottom of B_{v_1}. Due to Proposition 2, there are $(\mathcal{B}_d + 1)$ nodes on the bottom side of B_{v_1} which are connected to $\mathcal{N}_1$ by an I_{c_1}-avoiding path. The key point is that all these $(\mathcal{B}_d + 1)$ nodes are at the bottom of the same column in their respective copy of $\mathcal{G}_{\text{rec}}(\ell_1)$. Because of the wiring structure between B_{v_1} and B_{v_2} (vertical connections as depicted on Fig. 7), one of these bottom nodes is adjacent to a top-side node $\mathcal{N}_2$ of B_{v_2} belonging to an I_{c_2}-avoiding path. This allows the phenomenon to propagate in the whole graph $\mathcal{G}_{\text{rec}}(\ell_{k+1})$. The reader can find the details in Appendix C. A similar demonstration leads to the existence of an I-avoiding right–left path in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$ which achieves the demonstration of our theorem. (End of proof of Theorem 3.) □

We can simplify the formula for e_k from the previous theorem as follows. We have $\mathcal{B}_d \geq 2d - 1 > 1$. Therefore:

$$\sum_{i=1}^{k-1} (4\mathcal{B}_d)^i = \frac{(4\mathcal{B}_d)^k - 4\mathcal{B}_d}{4\mathcal{B}_d - 1}.$$

Using this equality, we obtain

$$e_k = 2(\mathcal{B}_d - 1)\mathcal{B}_d^{2k-1} + \frac{8\mathcal{B}_d^3 - 4\mathcal{B}_d^2 - \mathcal{B}_d}{4}\mathcal{B}_d^{2k-3}\frac{(4\mathcal{B}_d)^k - 4\mathcal{B}_d}{4\mathcal{B}_d - 1}.$$

Remark 14. We would like to point out the main difference with the recursive construction appearing in [28]. In that paper, the block B consists of $(\mathcal{B}_d + 1)^2$ copies

of $\mathcal{G}_{\text{rec}}(\ell_1)$. A similar analysis to what was performed above would show an identical formula for e_k with the exception that the factor $(8\mathcal{B}_d^3 - 4\mathcal{B}_d^2 - \mathcal{B}_d)$ would have been substituted by the larger value $2\mathcal{B}_d(\mathcal{B}_d^3 - 2\mathcal{B}_d^2 + 4\mathcal{B}_d - 1)$.

In [28], we studied the communication complexity of the recursive construction by upper bounding ℓ_k. Here, we can obtain a tighter bound directly from e_k. Indeed, the communication complexity of the protocol to t_k-privately compute the G-circuit using the previous admissible PDAG is $O(e_k N_g)$ group elements. We have

$$e_k \leq 2(\mathcal{B}_d - 1)\mathcal{B}_d^{2k-1} + 2\mathcal{B}_d^3\mathcal{B}_d^{2k-3}\frac{(4\mathcal{B}_d)^k}{3\mathcal{B}_d} \leq 2\mathcal{B}_d^{2k} + \frac{2^{2k+1}}{3}\mathcal{B}_d^{3k-1} \leq \frac{2^{2k+2}}{3}\mathcal{B}_d^{3k-1}.$$

It should be remembered that $n_k = (2d-1)^k$, in other words: $2^k = n_k^{1/\log_2(2d-1)}$. We can simplify the upper bound on e_k even further.

$$\begin{aligned} e_k &\leq \frac{4}{3} \times n_k^{2/\log_2(2d-1)} \times 2^{(2d-1)(3k-1)} \\ &\leq \frac{4}{3} \times n_k^{2/\log_2(2d-1)} \times \left(2^k\right)^{6d-3} \times 2^{1-2d} \\ &\leq \frac{1}{3 \times 2^{2d-3}} \times n_k^{6d-1/\log_2(2d-1)}. \end{aligned} \tag{3}$$

When d is a constant, the right hand side of inequality (3) is a polynomial in n_k.

Now, we generalize our result to any (t, n) where $t = O(n^{1-\epsilon})$ for any fixed positive ϵ. The class $O(n^{1-\epsilon})$ is the set of all functions f such that

$$\exists \tau_f > 0 \; \exists n_0 > 0 : \forall n \geq n_0 \quad f(n) \leq \tau_f n^{1-\epsilon}.$$

In our case, the function f is the privacy level t. Our main result is stated as follows.

Theorem 4. *For any fixed $\epsilon > 0$, for any fixed $\tau > 0$, there exists a constant $n_{\epsilon,\tau} \in \mathbb{N}$, such that for any $n \geq n_{\epsilon,\tau}$, if $t \leq \tau n^{1-\epsilon}$, then there exists a black-box t-private protocol for G-circuits with polynomial communication complexity $O(n^{6d-1/\log_2(2d-1)} N_g)$ using $O(n^{6d-1/\log_2(2d-1)} N_g)$ rounds where $d = 2^{\lceil 2/\epsilon \rceil - 1}$. Moreover, there is a deterministic polynomial-time algorithm to construct the protocol.*

Proof. We fix $\epsilon > 0$ and $\tau > 0$. We set $d = 2^{\lceil 2/\epsilon \rceil - 1}$ and $k = \lfloor \log_{(2d-1)} n \rfloor$. We have $d \geq 2$. If $n \geq 2d - 1$ then $k \geq 1$. In such a condition, we can apply Theorem 3 for the pair (k, d). There exists a t_k-private protocol for n_k participants to compute the G-circuit using $O(e_k N_g)$ group elements where t_k, n_k, e_k are defined as in Theorem 3. It is clear that the construction also t'-privately computes the G-circuit for any (t', n') such that $t' \leq t_k$ and $n' \geq n_k$. So, we only need to show $\tau n^{1-\epsilon} \leq t_k$, $n \geq n_k$ and $\ell_k = \mathcal{P}oly(n)$. Due to our choice of d and k, we have

$$n_k \leq (2d-1)^{\lfloor \log_{(2d-1)} n \rfloor} \leq (2d-1)^{\log_{(2d-1)} n} \leq n.$$

And:

$$t_k \geq d^{\lfloor \log_{(2d-1)} n \rfloor} - 1 \geq d^{\log_{(2d-1)} n - 1} - 1 \geq \frac{n^{\log_2 d / \log_2 (2d-1)}}{d} - 1 \geq \frac{n^{\log_2 d / \log_2 2d}}{d} - 1.$$

Since $d = 2^{\lceil 2/\epsilon \rceil - 1}$, we get

$$t_k \geq \frac{n^{\lceil 2/\epsilon \rceil - 1/\lceil 2/\epsilon \rceil}}{2^{\lceil 2/\epsilon \rceil - 1}} - 1 \geq \frac{n^{1-\epsilon/2}}{2^{\lceil 2/\epsilon \rceil - 1}} - 1 \geq \frac{n^{\epsilon/2}}{2^{\lceil 2/\epsilon \rceil - 1}} n^{1-\epsilon} - 1.$$

Since ϵ is a fixed positive constant, the mapping $n \mapsto \frac{n^{\epsilon/2}}{2^{\lceil 2/\epsilon \rceil - 1}}$ has an infinite limit. Therefore: $\exists \widetilde{n}_{\epsilon,\tau} > 0 : \forall n \geq \widetilde{n}_{\epsilon,\tau} \ \frac{n^{\epsilon/2}}{2^{\lceil 2/\epsilon \rceil - 1}} \geq \tau + \frac{1}{n^{1-\epsilon}}$.

Remember that we early required $n \geq 2d - 1$ in order to use Theorem 3. If we set $n_{\epsilon,\tau} := \max(2d - 1, \widetilde{n}_{\epsilon,\tau})$ then

$$\forall n \geq n_{\epsilon,\tau} \quad \begin{cases} n_k \leq n, \\ t_k \geq \tau n^{1-\epsilon} \geq t. \end{cases}$$

It remains to argue about e_k. Since $1 \leq n_k \leq n$ and $1 \leq \frac{6d-1}{\log_2(2d-1)}$, we have $e_k \leq \frac{1}{3 \times 2^{2d-3}} \times n^{6d-1/\log_2(2d-1)}$. Since d is independent from n, the value e_k is upper bounded by a polynomial in n.

Concerning the number of rounds, due to the recursive construction process, the grid $\mathcal{G}_{\text{rec}}(\ell_k)$ can be regarded as a grid $\mathcal{G}_{\text{rec}}(\ell_{k-1})$ where each node is a block B. More precisely, $\mathcal{G}_{\text{rec}}(\ell_k)$ can be decomposed into v_{k-1} blocks B. Let us focus on a single block B. Such a block is a $(4\mathcal{B}_d)$-copy over a $(\mathcal{B}_d + 1) \times (\mathcal{B}_d + 1)$ grid of $\mathcal{G}_{\text{rec}}(\ell_1)$. Assume that we number the nodes of this $(\mathcal{B}_d + 1) \times (\mathcal{B}_d + 1)$ grid from right to left and from top to bottom. In other words, the top right node is assigned the position $(1, 1)$ and the bottom left node is located at $(\mathcal{B}_d + 1, \mathcal{B}_d + 1)$. Due to the construction process, the only nodes of this grid occupied by a copy of $\mathcal{G}_{\text{rec}}(\ell_1)$ are on the extremal rows (topmost and bottommost) and columns (leftmost and rightmost) (see Fig. 6 as example). We identify a copy to its location and we pair these copies of $\mathcal{G}_{\text{rec}}(\ell_1)$ as follows.

$$\begin{aligned} &\big\{\mathfrak{P}_i := \big((1, i), (i, 1)\big) : i \in \{2, \ldots, \mathcal{B}_d + 1\}\big\} \\ &\quad \cup \big\{\mathfrak{P}'_i := \big((i, \mathcal{B}_d + 1), (\mathcal{B}_d + 1, i)\big) : i \in \{2, \ldots, \mathcal{B}_d\}\big\}. \end{aligned}$$

The only two copies without a match are located at $(1, 1)$ (top right) and $(\mathcal{B}_d + 1, \mathcal{B}_d + 1)$ (bottom left).

Due to the edge orientation, the computation over the block B works as follows. First, we perform the computation for the top right copy of $\mathcal{G}_{\text{rec}}(\ell_1)$. Second, when this computation if done, we simultaneously process the computation corresponding to the pair of copies $\mathfrak{P}_2$. Third, we sequentially repeat this for $\mathfrak{P}_3, \ldots, \mathfrak{P}_{\mathcal{B}_d+1}, \mathfrak{P}'_2, \ldots, \mathfrak{P}'_{\mathcal{B}_d}$. Finally, we execute the computation of the last remaining copy of $\mathcal{G}_{\text{rec}}(\ell_1)$ located at $(\mathcal{B}_d + 1, \mathcal{B}_d + 1)$ (bottom left).

Applying our reasoning from the proof of Corollary 1, the computation represented by a single copy of $\mathcal{G}_{\mathrm{rec}}(\ell_1)$ takes $2\ell_1 = 2\mathcal{B}_d$ rounds. As a consequence, with our approach, the number of rounds needed to perform all computations over a block B is $2\ell_1[1 + (\mathcal{B}_d + 1 - 2 + 1) + (\mathcal{B}_d - 2 + 1) + 1] = 2\mathcal{B}_d(2\mathcal{B}_d + 1)$ rounds.

The number of rounds needed for the computation represented by the PDAG $\mathcal{G}_{\mathrm{rec}}(\ell_k)$ is at most $2\mathcal{B}_d(2\mathcal{B}_d + 1)v_{k-1}$. Since $\mathcal{G}_{\mathrm{rec}}(\ell_k)$ has to be mirrored to obtain a PDAG for the n_k-party shared 2-product computation (see Sect. 3.2), the number of rounds needed to t_k-securely compute the G-circuit is at most

$$4\mathcal{B}_d(2\mathcal{B}_d + 1)v_{k-1}(N_g - 1).$$

We have

$$\begin{aligned} 4\mathcal{B}_d(2\mathcal{B}_d + 1)v_{k-1} &\leq 4\mathcal{B}_d(2\mathcal{B}_d + 1)2^{2k-4}\mathcal{B}_d^{3k-4} \\ &\leq \frac{3}{4}\mathcal{B}_d^{3k-2}\left(2^k\right)^2 \\ &\leq \frac{3}{4}n_k^{2/\log_2(2d-1)}\left(2^k\right)^{6d-3}2^{2-4d} \\ &\leq \frac{3}{2^{4d}}n_k^{6d-1/\log_2(2d-1)}. \end{aligned}$$

Since $n_k \leq n$ and d is constant, we get our result.

3.4. *Probabilistic Construction*

It is natural to ask whether the exponentially large sharing parameter $\ell = \binom{n}{t}$ can be reduced to polynomial size while allowing t to be a constant fraction of n. In this section, we will provide a positive answer to this question, in fact our construction will allow an optimal $t < n/2$, but only probabilistically.

For our construction, we use the 'probabilistic method' [1]. Namely, we choose the color of each node in the square graph $\mathcal{G}_{\mathrm{tri}}(\ell, \ell)$ independently and uniformly at random from $[n]$. Although there is a finite error probability δ that such a random n-coloring will not be weakly t-reliable, we show that for $t < n/2$ with sufficiently large sharing parameter ℓ (still *polynomial* in n), the error probability δ can be made arbitrarily small. Moreover, δ decreases exponentially fast with ℓ, so δ can be easily made negligible. We also show that for $t = n/(2 + \epsilon)$ for any constant $\epsilon > 0$, it is sufficient to take a linear sharing parameter $\ell = O(n)$ for arbitrarily small constant error probability δ.

Remark 15. Although our results allow one to efficiently generate colorings for $\mathcal{G}_{\mathrm{tri}}(\ell, \ell)$ which are weakly t-reliable except for a negligible error probability, it is natural to ask whether one can efficiently generate colorings which we are *certain* to be weakly t-reliable. In this connection, we note that for any *given* t-collusion I, and a coloring C, there exists an efficient algorithm (with run time linear in the number of nodes ℓ^2) to verify that the colored 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

Algorithm 4 Coloring C_{rand}

Input: A grid $\mathcal{G}_{\text{tri}}(\ell, \ell)$.

1. For each $(i, j) \in [\ell] \times [\ell]$, choose the color $C(i, j)$ of node (i, j) independently and uniformly at random from $[n]$.

Output: A n-coloring of the grid.

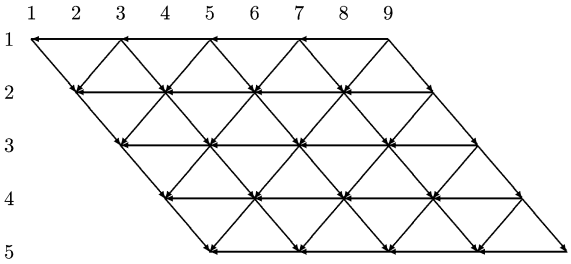


Fig. 8. The triangle $\mathcal{T}(5)$.

exponential in t. We do not know whether there is an efficient algorithm to verify that a given a given coloring C of $\mathcal{G}_{\text{tri}}(\ell, \ell)$ is weakly t-reliable.

We use the random coloring C_{rand} for the grid $\mathcal{G}_{\text{tri}}(\ell, \ell)$ described as Algorithm 4.

Our analysis will be based on percolation theory. We first introduce the following definition which is illustrated in Fig. 8.

Definition 7. The triangular lattice of depth ℓ denoted $\mathcal{T}(\ell)$ is a directed graph drawn over an $\ell \times (3\ell - 2)$ grid such that

- [horizontal edges] for $i \in [\ell]$ and for $j \in [\ell - 1]$, there is a directed edge from node $(i, i + 2j)$ to $(i, i + 2(j - 1))$,
- [right downwards edges] for $i \in [\ell - 1]$ and for $j \in \{0, \ldots, \ell - 1\}$, there is a directed edge from node $(i, i + 2j)$ to node $(i + 1, i + 2j + 1)$,
- [left downwards edges] for $i \in [\ell - 1]$ and for $j \in [\ell - 1]$, there is a directed edge from node $(i, i + 2j)$ to node $(i + 1, i + 2j - 1)$.

Proposition 3. *For any positive integer ℓ, we have a graph isomorphism between $\mathcal{G}_{\text{tri}}(\ell, \ell)$ and $\mathcal{T}(\ell)$.*

Theorem 5. *For any $\epsilon > 0$ and $\delta > 0$, there exists a constant c_ϵ such that if $t \leq \frac{n}{2+\epsilon}$ and $\ell \geq c_\epsilon(n + \log \delta^{-1})$, then there exists a weakly t-reliable n-coloring for $\mathcal{G}_{\text{tri}}(\ell, \ell)$, and Algorithm 4 returns such a coloring with probability at least $1 - \delta$. Furthermore, if $n = 2t + 1$, then for any $\nu > 0$ and $\delta > 0$, there exists a constant c_ν such that for $\ell \geq c_\nu n^{91/36+\nu}(n + \log \delta^{-1})$, Algorithm 4 returns a weakly t-reliable n-coloring for $\mathcal{G}_{\text{tri}}(\ell, \ell)$ with probability at least $1 - \delta$.*

Proof. The demonstration of this theorem requires some results on percolation theory which can be found in Appendix D.

We prove that the coloring C_{rand} will work with high probability. Let $t_\epsilon = \lfloor \frac{n}{2+\epsilon} \rfloor$ where $\lfloor \cdot \rfloor$ denotes the floor function, and ϵ is either a constant (for the first part of the theorem), or $\epsilon = 2/(n-1)$ (for the second part). Instead of considering the probability that C_{rand} is a weakly t_ϵ-reliable n-coloring for $\mathcal{G}_{\text{tri}}(\ell, \ell)$, we study the complementary event. A suitable value for ℓ will be given at the end of this demonstration.

The coloring C_{rand} is called *bad* if there exists a color set $I \subset [n]$ with $|I| = t_\epsilon$, such that either there are no I-avoiding top-bottom paths or there are no I-avoiding right–left paths. By the union bound, we obtain the following upper bound on $\Pr(C_{\text{rand}}$ is bad):

$$\begin{aligned} &2\Pr\big(\exists I \subset [n], |I| = t_\epsilon, \text{there are no } I\text{-avoiding top-bottom paths in } \mathcal{G}_{\text{tri}}(\ell, \ell)\big) \\ &\quad \leq 2 \sum_{I \subset [n], |I| = t_\epsilon} \Pr\big(\text{there are no } I\text{-avoiding top-bottom paths in } \mathcal{G}_{\text{tri}}(\ell, \ell)\big). \end{aligned} \tag{4}$$

The factor 2 in inequality (4) comes from the fact the top-bottom probability is equal to the right–left probability due to the symmetry of the grid $\mathcal{G}_{\text{tri}}(\ell, \ell)$ and the coloring C_{rand}.

Next, we demonstrate that for a fixed color set $I \subset [n]$ with $|I| = t_\epsilon$, the probability that there are no I-avoiding top-bottom paths in C_{rand} is exponentially small. Let us fix the color set I. We call a vertex *closed* if its color belongs to I. Otherwise, the vertex is called *open*. The random coloring C_{rand} of each vertex is equivalent to opening it independently and randomly with probability $p := 1 - \frac{t_\epsilon}{n}$. An I-avoiding path is simply an *open path*. Therefore, we get

$$\begin{aligned} &\Pr\big(\text{there are no } I\text{-avoiding top-bottom paths in } \mathcal{G}_{\text{tri}}(\ell, \ell)\big) \\ &\quad = \Pr_p\big(\text{there are no open top-bottom paths in } \mathcal{G}_{\text{tri}}(\ell, \ell)\big) \\ &\quad = 1 - \Pr_p\big(\text{there is an open top-bottom path in } \mathcal{G}_{\text{tri}}(\ell, \ell)\big). \end{aligned} \tag{5}$$

When we combine Lemma 5 (Appendix D), Proposition 3 and equality (5), we obtain

$$\begin{aligned} &\Pr\big(\text{there is no } I\text{-avoiding top-bottom path in } \mathcal{G}_{\text{tri}}(\ell, \ell)\big) \\ &\quad = \Pr_p\big(\text{there is a closed right–left path in } \mathcal{T}(\ell)\big) \\ &\quad = \Pr_{1-p}\big(\text{there is an open right–left path in } \mathcal{T}(\ell)\big). \end{aligned} \tag{6}$$

In equality (6), $\Pr_{1-p}(\cdot)$ means that we open each vertex with probability $1 - p$. In our case, we have $1 - p = \frac{t_\epsilon}{n} \leq \frac{1}{2+\epsilon} < p_c^s(\mathcal{T}(\ell))$ where $p_c^s(\mathcal{T}(\ell))$ denotes the critical probability of site percolation for $\mathcal{T}(\ell)$. Using Lemma 7 (Appendix D), we get

$$\Pr_{1-p}\big(\text{there is an open right–left path in } \mathcal{T}(\ell)\big) \leq \ell \Pr_{1-p}\big(0 \xrightarrow{\ell-1}\big) \leq \ell e^{-c(\ell-1)}. \tag{7}$$

The first inequality is due to the fact that any right–left path has length at least $(\ell - 1)$ in $\mathcal{T}(\ell)$. Combining (4) to (7), we obtain

$$\Pr(C_{\text{rand}} \text{ is bad}) \leq 2\binom{n}{t_\epsilon} \ell e^{-c(\ell-1)}.$$

Thus, if we choose $\ell := c_\epsilon(n + \log \delta^{-1})$ for some large enough constant c_ϵ, we have $\Pr(C_{\text{rand}} \text{ is bad}) \leq \delta$, as required. This proves the first part of the theorem for constant $\epsilon > 0$.

For the second part of the theorem, with $\epsilon = 2/(n-1)$, we need Lemma 8 (Appendix D) representing a quantitative version of Lemma 7. As in the proof of the first part of the theorem, we apply this result with opening probability $1 - p = \frac{t_\epsilon}{n} = \frac{1}{2+\epsilon} < p_c^s(\mathcal{T}(\ell))$ with $\epsilon = 2/(n-1)$ approaching zero with increasing n. Note that $1 - p \geq 1/3$. Using Lemma 8 and the parameters from Theorem 6 (Appendix D), we get

$$\Pr_{1-p}\big(\text{there is an open right–left path in } \mathcal{T}(\ell)\big) \leq \ell \Pr_{1-p}\big(0 \xrightarrow{\ell-1}\big) \leq \ell e^{-\lfloor(\ell-1)/r(1-p)\rfloor}$$

with $r(1-p) = c\xi(1-p)\sqrt{\chi(1-p)} = c(p-1/2)^{-(4/3+43/36)+o(1)}$ as p tends to $1/2$. Using $p - 1/2 = 1/(2n)$, we get $r(n) = c' n^{91/36+o(1)}$ for some absolute constant c'. Following the same union bound argument as in the first part of the theorem, this leads to a coloring failure probability at most δ if $\ell \geq r(n)(n + \log(\delta^{-1}))$, as claimed. □

Combining Theorem 5 and with Remark 11, Lemmas 1 and 2, we get the following.

Corollary 2. *For any $\delta > 0$, if $t < n/2$, we can construct a probabilistic algorithm, with run-time polynomial in n and $\log(\delta^{-1})$, which outputs a black-box protocol $\prod$ for G-circuits such that the communication complexity of $\prod$ is $O(n^{5.056}(n + \log \delta^{-1})^2 N_g)$ group elements, the protocol requires $O(n^{2.528}(n + \log \delta^{-1})N_g)$ rounds and the probability that $\prod$ is not t-private is at most δ. Moreover, if $t \leq n/(2+\epsilon)$ for some constant $\epsilon > 0$, the communication complexity is $O((n + \log \delta^{-1})^2 N_g)$ group elements and the protocol needs $O((n + \log \delta^{-1})N_g)$ rounds.*

4. Conclusion

In this paper, we showed how to design black-box t-private protocols for performing the secure evaluation of an arbitrary circuit over a finite group. This result was obtained by reducing the cryptographic problem of performing MPC over (non-Abelian) groups to a combinatorial graph coloring problem, using tools from communication security [12]. We then gave two deterministic solutions to the problem. Our first construction achieves optimal resilience $t < n/2$ but it has exponential complexity. Our second construction is a recursive technique which demonstrates the existence of a t-secure protocol to compute G-circuits using polynomial communication complexity for any $t \in O(n^{1-\epsilon})$ where ϵ is any positive constant. We also present a probabilistic construction with communication complexity polynomial in n and achieving an optimal collusion security $t < n/2$. All these results, holding for the passive adversarial model, are summarized in Table 1.

Open Problems Our work raises some interesting combinatorial questions. For example, what is the largest collusion resistance achievable with an admissible PDAG of size polynomial in n, and what kind of PDAG achieves this optimum? Theorem 5 partly answers this question. Indeed, it states that optimal collusion resistance ($t = (n-1)/2$)

Type of construction	Coloring	Communication complexity	Number of adversaries	Number of rounds
Deterministic	C_{comb}	$O(\binom{2t+1}{t}^2 N_g)$	$t \leq \lceil \frac{n}{2} \rceil - 1$ (optimal)	$O(\binom{2t+1}{t} N_g)$
	Recursion from C_{comb}	$O(n^{6d-1/\log_2(2d-1)} N_g)$ where $d = 2^{\lceil 2/\epsilon \rceil - 1}$	$t = O(n^{1-\epsilon})$ (ϵ constant)	$O(n^{6d-1/\log_2(2d-1)} N_g)$ where $d = 2^{\lceil 2/\epsilon \rceil - 1}$
Probabilistic (failure δ)	C_{rand}	$O(n^{5.056}(n + \log(\delta^{-1}))^2 N_g)$	$t \leq \lceil \frac{n}{2} \rceil - 1$ (optimal)	$O(n^{2.528}(n + \log(\delta^{-1}))N_g)$
	C_{rand}	$O((n + \log(\delta^{-1}))^2 N_g)$	$t \leq \frac{n}{2+\epsilon}$ (ϵ constant)	$O((n + \log(\delta^{-1}))N_g)$

Table 1. Comparative summary of protocols for computing G-circuits in the passive adversarial model.

is achievable with a polynomial size admissible PDAG at the cost of using a coloring which is not t-reliable with probability $\delta > 0$. There are also interesting cryptographic questions. First, can the communication complexity of our protocols be reduced further? Second, does there exist an efficient (running-time polynomial in n) *deterministic* algorithm to generate a weakly t-reliable n-coloring of $\mathcal{G}_{\text{tri}}(\ell, \ell)$ (or some other admissible PDAG) given n, t as input for any $t < n$? Can such a coloring be generated recursively as our second deterministic construction? During our research, we obtained several suitable pairs of colorings/graphs for small values of n and ℓ. Our most representative outcomes can be found in Appendix E. Although we were not able to generalize those examples, these structures might be of interest for future investigations in this area.

Since all our constructions only provide security in the passive adversary case, one may wonder if these techniques can be extended to deal with active adversaries.

In MPC, the number of rounds is an important efficiency measure. One may ask whether one can design an admissible PDAG implying a protocol running in a constant number of rounds such as [10]. Unfortunately, this is impossible to achieve if we want optimal security (i.e. when $n = 2t + 1$). Indeed, a constant number of rounds would imply the graph diameter to be constant. However, if the diameter is less than $n/2$ then there is a path containing less than $n/2$ colors which can disconnect the graph (top-bottom/left-right wise). More generally, to ensure the security against t adversaries, the graph diameter must be at least $t + 1$. As a consequence, one needs to tackle the problem of MPC over a non-Abelian group in a different way to get a protocol with a constant number of rounds (if possible).

The reader may notice that associativity is a fundamental property used in our protocols. Indeed, an invariant property is the fact that, at every row of our PDAGs, the left-to-right product of the ℓ shares is constant. This implies that our approach cannot be used to perform MPC over structures for which the law of composition is not associative such as quasigroups. In this situation, one may wonder what technique should be used to perform secure MPC.

Acknowledgements

The authors would like to thank the reviewers for their suggestions to improve the quality of this paper. We 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 MPC via the work of Barrington [3]. The work of the first author was funded by NSF ANI-0087641, EPSRC EP/C538285/1. Parts of his research were done while being BT Chair of Information Security and during visits funded by the National Natural Science Foundation of China grant 60553001, the National Basic Research Program of China grants 2007CB807900 and 2007CB807901 as well as by Australian Research Council grants DP0663452 and DP0987734. The work of the second and third authors was supported by Australian Research Council grants DP0451484, DP0558773, DP0663452, DP0665035, and DP0987734. The work of the third author was also supported in part by a Macquarie University Research Fellowship (MURF) and ARC Discovery Grant DP110100628. The sixth author was supported in part by the Australian Research Council under ARC Discovery Project DP0665035, the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03 and the Singapore Ministry of Education under Research Grant T206B2204. The fourth, fifth and seventh authors are supported by the National Natural Science Foundation of China grants 60553001, 61033001, 61061130540, 61073174 and the National Basic Research Program of China grants 2007CB807900 and 2007CB807901. The work of the fourth author is also funded by the National Natural Science Foundation of China under grant 60603005, 60621062 and Tsinghua University Initiative Scientific Research Program 2009THZ02120. The fourth, fifth and seventh authors also acknowledge support from the Danish National Research Foundation and the National Natural Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation (CTIC) within which part of this work was performed. The work of the fifth author is also financed by the National Natural Science Foundation of China under the International Young Scientists program (grants 61050110147 and 61150110344).

Appendix A. Proof of Lemma 1

The correctness of $\prod_S^{(x)}$ and $\prod_S^{(y)}$ implies that the protocol output at each internal Mult node α in C' is an ℓ-of-ℓ sharing of the product of the two values whose sharings are the labels of the incoming edges to α. If α is a CMult node, it is clear that its protocol value is an $\mathcal{O}_x$ sharing of $\alpha x \beta$, where x is the $\mathcal{O}_x$-shared input value. It follows easily by induction that the protocol value computed at the output node of C' is an ℓ-of-ℓ sharing of $f_{C'}(x_1, \ldots, x_n)$, establishing correctness.

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 $y = f_{C'}(x_1 \cdots x_n)$, the algorithm S runs as follows.

1. For each input node i of C', 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 $P_{\mathcal{O}_x(j)}$ for $j \in [\ell] \setminus \{j_x^*\}$, where $j_x^* \in [\ell]$ (which depends on I) is the share indices which are *not* simulated/input by/to simulator $\mathsf{S}_{\prod_S}$ of subprotocols $\prod_S^{(x)}$ and $\prod_S^{(y)}$ (see Definition 2). S labels leaf node i with $\{s_{x_i}(j)\}_{j \in [\ell] \setminus \{j_x^*\}}$.

2. For each internal node i of C', if i is a CMult node with input shares $\{s_x(j)\}_{j\in[\ell]\setminus\{j_x^*\}}$, S computes and labels i with the output shares (except the j_x^*th) $\{s_z^{(i)}(j)\}_{j\in[\ell]\setminus\{j_x^*\}}$ for node i by following the protocol (i.e. $s_z^{(i)}(1)=\alpha\cdot s_x^{(i)}(1)$, $s_z^{(i)}(\ell)=\alpha\cdot s_x^{(i)}(\ell)\cdot\beta$ and $s_z^{(i)}(j)=s_x^{(i)}(j)$ for $j\notin\{1,\ell\}$).

   If i is a Mult node with input shares $\{s_x(j)\}_{j\in[\ell]\setminus\{j_x^*\}}$, and $\{s_y(j)\}_{j\in[\ell]\setminus\{j_x^*\}}$, S first runs simulator $\mathsf{S}_{\prod_S^{(y)}}$ of subprotocol $\prod_S^{(y)}$ on input $(I,\{s_y(j)\}_{j\in[\ell]\setminus\{j_x^*\}}$, $\{s_1(j)\}_{j\in[\ell]\setminus\{j_y^*\}})$, where $\{s_1(j)\}_{j\in[\ell]\setminus\{j_y^*\}}$, is a random sharing of 1_{S_5}. Simulator $\mathsf{S}_{\prod_S^{(y)}}$ outputs a simulation of the internal view of I in the Convert subprotocol run at node i, along with simulated output shares for Convert (except the j_y^*th) $\{s'_y(j)\}_{j\in[\ell]\setminus\{j_y^*\}}$. Then S runs simulator $\mathsf{S}_{\prod_S^{(x)}}$ of subprotocol $\prod_S^{(x)}$ on input $(I,\{s_x(j)\}_{j\in[\ell]\setminus\{j_x^*\}},\{s'_y(j)\}_{j\in[\ell]\setminus\{j_y^*\}})$. Simulator $\mathsf{S}_{\prod_S^{(x)}}$ outputs a simulation of the internal view of I in the $\prod_S^{(x)}$ subprotocol run at node i, along with simulated output shares $\{s_z^{(i)}(j)\}_{j\in[\ell]\setminus\{j_x^*\}}$ of $\prod_S^{(x)}$. S labels node i of C' with $\{s_z^{(i)}(j)\}_{j\in[\ell]\setminus\{j_x^*\}}$.
3. After labeling all nodes of C', S has a complete simulation of the view of I in $\prod(C',\prod_S^{(x)})$, except for the j_x^*th share $s_z^*(j_x^*)$ of output node label, broadcast by party $P_{\mathcal{O}_z(j_x^*)}$ in the last round. However, we know that the output node shares satisfy $s_z^*(1)\cdots s_z^*(\ell)=y$. Accordingly, from the known value of y and the simulated shares $\{s_z(j)\}_{j\in[\ell]\setminus\{j^*\}}$, $s_z^*(j^*)$ can be uniquely determined and simulated by S.

For each internal node of C', the perfect simulation of the protocol view and output at the node given the view so far follows inductively from the strong t-privacy of $\prod_S^{(x)}$ and $\prod_S^{(y)}$. This completes the proof of Lemma 1.

Appendix B. Proof of Lemma 2

The correctness of $\prod_S(\mathcal{G},C)$ is immediate from the fact that, due to the ordering condition (1) in Definition 4, 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 $\prod_S(\mathcal{G},C)$, let $I\subseteq[n]$ denote an arbitrary t-color 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 $\mathsf{S}_{\prod_S}$ 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 $\prod_S$ 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 $\mathsf{S}_{\prod_S}$ contain the view of I, and $\mathsf{S}_{\prod_S}$ 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 $PATH_y$ which are incoming edges to a node on $PATH_x$ or $PATH_y$. We show that, in the real subprotocol $\prod_S$, 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 $\prod_S$ 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

$$\alpha_1 r_i \alpha_2 = \alpha_3, \tag{B.1}$$

  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 Fig. 9 for an example.

Let us first consider the nodes on $PATH_y$. By Proposition 1, we may assume (without changing the protocol distribution) that in the real subprotocol $\prod_S$, 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 (B.1) is trivially satisfied, since these equations have the form $r_i = \alpha$, where r_i is new and α is fixed by V.

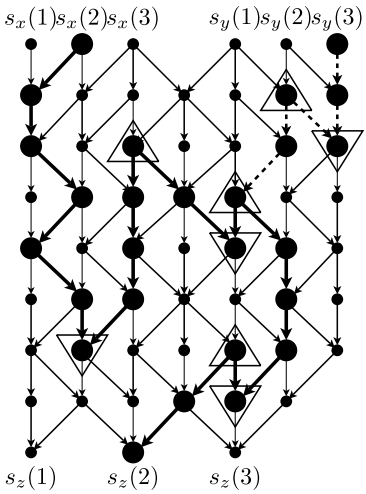


Fig. 9. Example of an admissible PDAG with sharing parameter $\ell = 3$ (node colors are not indicated). For a given collusion I, an example I-avoiding path $PATH_x$ is shown in *heavy black*, and an example I-avoiding path $PATH_y$ (until the meeting with $PATH_x$) is shown in *dashed black*. Minimum path nodes are marked by a *triangle with a horizontal edge above* the node, whereas maximum path nodes are marked by a *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.

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 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 labeled with new random elements r_i. We take the corresponding view edge equations as the next ones in our ordering, again trivially satisfying (B.1). Of the two outgoing edges on the path from the jth maximum node, one of them goes to an *earlier* node on the path (toward the jth 'minimum' path node), and we may assume that this edge is labeled 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

$$v_{j,j} = \alpha_1 r_j^* \alpha_2, \tag{B.2}$$

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 the $(j+1)$th minimum is of the form

$$v_{j,j+1} = \beta_1 \left(r_j^*\right)^{-1} \beta_2, \tag{B.3}$$

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 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 (B.1), and it remains to consider 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

$$\gamma_1 v_{j-1,j}^{\pm 1} \gamma_2 v_{j,j}^{\pm 1} \gamma_3 = \gamma_4, \tag{B.4}$$

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. toward 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 (B.2), the label $v_{1,1}$ on the incoming path edge to the first minimum node from the next node on the path (i.e. toward 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 (B.4), 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 (B.1) is satisfied for $j = 1$.

For $j > 1$, using (B.3), we have $v_{j-1,j} = \beta_1 (r_{j-1}^*)^{-1} \beta_2$ with β_1, β_2 fixed by V, while using (B.2), 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 (B.4), 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 (B.1) is also satisfied for $1 < j \leq k$.

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 (B.4) 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 (B.1) 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, which completes the proof of the Lemma 2.

Appendix C. Connection of Color Avoiding Paths

It was shown in the proof of Theorem 3 that each block B_{c_i} had $\mathcal{B}_d$ horizontal and $\mathcal{B}_d$ vertical I_{c_i}-avoiding paths. In this Appendix, we show how to construct a I-avoiding top-bottom path in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$. Our path will start at the top of B_{v_1} and ends at the bottom of B_{v_m}.

Every grid from the family $(\mathcal{G}_{\text{rec}}(\ell_\lambda))_{\lambda \geq 1}$ is a square grid. Thus, the sequence of blocks $B_{v_1}, \ldots, B_{v_m}$ in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$ is determined by the position of B_{v_1} as well as the m-tuple of letters from $\{\mathfrak{L}, \mathfrak{R}, \mathfrak{T}, \mathfrak{B}\}$ ($\mathfrak{L}$eft, $\mathfrak{R}$ight, $\mathfrak{T}$op, $\mathfrak{B}$ottom) indicating the output

side of the block B_{v_i} for $i \in [m]$. Note that the last letter of the tuple is always $\mathfrak{B}$ since the I-avoiding top-bottom path ends at the bottom of B_{v_m}.

This tuple has the property the two consecutive letters cannot be opposite to each other (i.e., one cannot have $(\mathfrak{L}, \mathfrak{R})$, $(\mathfrak{R}, \mathfrak{L})$, $(\mathfrak{T}, \mathfrak{B})$ or $(\mathfrak{B}, \mathfrak{T})$). This means that you leave a block on a different side that you entered it. The reader can check the correctness of this claim by a simple recursive process on the parameter k. This property is trivially true for $k = 1$ since $\mathcal{G}_{\text{rec}}(\ell_1) = \mathcal{G}_{\text{tri}}(\ell_1)$. The recursion follows from the path construction that we will design below.

Proposition 4. *Let i be any element of $[m]$. Assume that $\mathcal{N}$ is any node on a side of B_{v_i} belonging to a I_{c_i}-avoiding straight line path. For each other side $\mathfrak{S}_i$ of B_{v_i}, we can construct a I_{c_i}-avoiding path from $\mathcal{N}$ to any of the $(\mathcal{B}_d + 1)$ nodes on $\mathfrak{S}_i$ belonging to a I_{c_i}-avoiding straight line path.*

Proof. We only provide a proof when $\mathcal{N}$ is on the top side of B_{v_i} (the three other cases are similar). The three possible output sides are $\mathfrak{B}$, $\mathfrak{L}$ and $\mathfrak{R}$. The block B_{v_i} is a-$(\mathcal{B}_d + 1) \times (\mathcal{B}_d + 1)$-copy of the original grid $\mathcal{G}_{\text{rec}}(\ell_1)$. Thus, B_{v_i} can be treated as a $(\mathcal{B}_d + 1) \times (\mathcal{B}_d + 1)$ array of grids $\mathcal{G}_{\text{rec}}(\ell_1)$. Based on this observation, we will use the terminology *grid-row* (respectively, *grid-column*) to denote a set of $\mathcal{B}_d + 1$ horizontal (respectively, vertical) grids $\mathcal{G}_{\text{rec}}(\ell_1)$ in B_{v_i}.

1. $\mathfrak{S}_i = \mathfrak{B}$. The vertical I_{c_i}-avoiding path starting at node $\mathcal{N}$ intersects the *horizontal* I_{c_i}-avoiding path located within the bottom grid-row of B_{v_i} at node $\mathcal{I}$. That horizontal path intersects each of the $\mathcal{B}_d + 1$ *vertical* I_{c_i}-avoiding paths (one within each grid-column) at $\mathcal{I}_1, \ldots, \mathcal{I}_{\mathcal{B}_d+1}$. Note that $\mathcal{I} = \mathcal{I}_\mu$ for some $\mu \in [\mathcal{B}_d + 1]$. Once we are at one of the $\mathcal{I}_j$'s, we simply go vertically downwards to the node $\mathcal{N}'_j$ located at the bottom side of the block B_{v_i}.

Thus, we can construct a path from $\mathcal{N}$ to each of the $\mathcal{B}_d + 1$ output nodes on the bottom side of B_{v_j} belonging to the vertical I_{c_i}-avoiding paths. Those paths are $(\mathcal{N}, \mathcal{I}, \mathcal{I}_j, \mathcal{N}'_j)$ for $j \in [\mathcal{B}_d + 1]$.

2. $\mathfrak{S}_i = \mathfrak{R}$. The vertical I_{c_i}-avoiding path starting at node $\mathcal{N}$ intersects the *horizontal* I_{c_i}-avoiding path located within the top grid-row of B_{v_i} at node $\mathcal{I}$. That horizontal path intersects the *vertical* I_{c_i}-avoiding path located within the rightmost grid-column of B_{v_i} at node $\widetilde{\mathcal{I}}$. This vertical path intersects each of the $\mathcal{B}_d + 1$ *horizontal* I_{c_i}-avoiding paths (one within each grid-row) at $\widetilde{\mathcal{I}}_1, \ldots, \widetilde{\mathcal{I}}_{\mathcal{B}_d+1}$. As before, we get $\widetilde{\mathcal{I}} = \widetilde{\mathcal{I}}_\mu$ for some $\mu \in [\mathcal{B}_d + 1]$. Once we are at one of the $\widetilde{\mathcal{I}}_j$'s, we horizontally go rightwards to the node $\mathcal{N}'_j$ located on the right hand side of the block B_{v_i}.

Thus, we can construct a path from $\mathcal{N}$ to each of the $\mathcal{B}_d + 1$ output nodes on the right hand side of B_{v_j} belonging to the horizontal I_{c_i}-avoiding paths. Those paths are $(\mathcal{N}, \mathcal{I}, \widetilde{\mathcal{I}}, \widetilde{\mathcal{I}}_j, \mathcal{N}'_j)$ for $j \in [\mathcal{B}_d + 1]$.

3. $\mathfrak{S}_i = \mathfrak{L}$. This is analogous to the previous case. □

We can finally construct a I-avoiding top-bottom path in $\mathcal{G}_{\text{rec}}(\ell_{k+1})$. We denote the m-tuple of output sides as $(\mathfrak{S}_1, \ldots, \mathfrak{S}_m)$. As previously said, we have $\mathfrak{S}_m = \mathfrak{B}$.

We start at *any* node N_1 located on the top side of B_{v_1} and on a vertical I_{c_1}-avoiding path. Using Proposition 4, we can connect N_1 to any of the $\mathcal{B}_d + 1$ nodes on side $\mathfrak{S}_1$ of B_{v_1} using a I_{c_1}-avoiding path. An important remark is that each block of the whole

grid $\mathcal{G}_{\text{rec}}(\ell_{k+1})$ is a set of $(\mathcal{B}_d + 1) \times (\mathcal{B}_d + 1)$ identical copies of $\mathcal{G}_{\text{rec}}(\ell_1)$ (including the coloring). As a consequence, these $\mathcal{B}_d + 1$ nodes have the same location in their respective copies of $\mathcal{G}_{\text{rec}}(\ell_1)$. Given the connection process between any pair of blocks within $\mathcal{G}_{\text{rec}}(\ell_{k+1})$, one of these $\mathcal{B}_d + 1$ nodes must be connected to a node N_2 from block B_{v_2} belonging to a I_{c_2}-avoiding straight line path. Similarly, N_2 is connected via a I_{c_2}-avoiding path in B_{v_2} to a node N_3 from B_{v_3} belonging to a I_{c_3}-avoiding straight line path. If we repeat this process for each of the remaining blocks, we obtain a set of $m - 1$ nodes $N_1, \ldots, N_{m-1}$. The last node N_{m-1} can be connected to a node N_m on the bottom side of B_{v_m} using a I_{c_m}-avoiding path. Thus, N_1 (top side of $\mathcal{G}_{\text{rec}}(\ell_{k+1})$) is connected to N_m (bottom side of $\mathcal{G}_{\text{rec}}(\ell_{k+1})$) using a I-avoiding path which achieves the demonstration of our theorem.

Remark 16. As claimed above, this construction involves that the two consecutive side letters of the m-tuple cannot be opposite to each other.

Appendix D. Selected Results on Percolation Theory

In this Appendix, we recall some existing results on percolation theory and we present some additional properties. These are needed to demonstrate Theorem 5.

Lemma 5 [6]. *The triangular lattice $\mathcal{T}(\ell)$ (Definition 7) has the following property*:

$$\Pr_p\big(\textit{there is an open top-bottom path in } \mathcal{T}(\ell)\big) + \Pr_p\big(\textit{there is a closed right–left path in } \mathcal{T}(\ell)\big) = 1.$$

Lemma 6 [21]. *Let T be the triangular lattice in the plane. Then, the critical probability of site percolation $p_c^s(T)$ is equal to $\frac{1}{2}$.*

When the open probability is less than the critical probability, the percolation has the following properties (see for example Chap. 4, Theorem 9 in [6]).

Lemma 7 [19]. *If $p < p_c^s(T)$, then there is a constant $c = c(p)$,*

$$\Pr_p\big(0 \xrightarrow{n}\big) < e^{-cn},$$

where $\{x \xrightarrow{n}\}$ is the event that there is an open path from x to a point in $S_n(x)$ with $S_n(x) := \{y : d(x, y) = n\}$ and $d(x, y)$ denotes the distance between x and y.

Remark 17. The value 0 from Lemma 7 represent the zero element of $\mathbb{Z} \times \mathbb{Z}$ when the graph is represented as a lattice over that set. In the case of the triangular lattice depicted as Fig. 8, the value 0 can be identified to the node $(1, 1)$.

Lemma 8. *Suppose* $1/3 \leq p < p_c^s(T)$, *and define random variable* N *as the cardinality of the open cluster of nodes* C_0 *in* T *containing the origin. Let* $\chi(p)$ *and* $\xi(p)$ *denote the following expected values*:

$$\chi(p) = \mathbb{E}_p(N),$$

and

$$\xi(p) = \big(\mathbb{E}_p(L)/\chi(p)\big)^{1/2}, \quad \textit{where } L = \sum_{y \in C_0} \|y\|_2^2.$$

Here, $\|y\|_2$ *denotes the Euclidean distance of node* y *from the origin. Then, there exists an absolute constant* c *such that*

$$\Pr_p\big(0 \xrightarrow{n}\big) < e^{-\lfloor n/r(p) \rfloor},$$

with $r(p) = c\xi(p)\sqrt{\chi(p)}$, *and* $\{x \xrightarrow{n}\}$ *is defined as in Lemma 7.*

Proof. For $r \geq 1$, let B_r^+ denote the set of nodes of T at graph distance at most r from the origin. Also, define the random variable N_r^+ to be the number of nodes in T at graph distance exactly r from the origin, which may be reached by an open path in B_r^+ starting at a node neighboring the origin. It is shown in Sect. 4.4, Lemma 8 of [6] that, for any $r \geq 1$, any $n \geq 1$ and any real $\gamma < 1$, if $\mathbb{E}_p(N_r^+) \leq \gamma$ then we have $\Pr_p(0 \xrightarrow{n}) \leq p\gamma^{\lfloor n/r \rfloor}$. Thus our lemma immediately follows once we show that $\mathbb{E}_p(N_r^+) \leq e^{-1}$ for $r \geq r(p)$.

Let N_r be defined equal to N_r^+ except that we set $N_r = 0$ if the origin node is closed. Notice that $N_r = N_r^+ X$, where $X = 1$ if the origin is open, and $X = 0$ otherwise. Since the random variables N_r^+ and X are independent, we have $\mathbb{E}_p(N_r) = p\mathbb{E}_p(N_r^+)$. Hence, if $p \geq 1/3$, it suffices to show that $\mathbb{E}_p(N_r) \leq e^{-1}/3$ for $r \geq r(p)$.

For this, we observe first that for any node x, we have $\|x\|_2 \geq d(0,x)/\sqrt{3}$. This follows by elementary geometry, where we assume the standard scaling of the lattice T (see [6], Sect. 5.3) in which the triangles are equilateral with side length 1 and the points $(0,0)$ and $(1,0)$ are in T. With this representation, a basis for T is formed by vectors $b_1 = (1,0)$ and $b_2 = (1/2, \sqrt{3}/2)$. Therefore, each node x among the N_r nodes at graph distance r from the origin, is at Euclidean distance at least $r/\sqrt{3}$ from the origin, and is in the open cluster C_0 containing the origin. It follows that $L \geq N_r(r/\sqrt{3})^2$, and hence $\mathbb{E}_p(N_r) \leq (3/r^2)\mathbb{E}_p(L)$. Using $\mathbb{E}_p(L) = \xi(p)^2\chi(p)$, we conclude that it is sufficient to take $r \geq 3\sqrt{e\chi(p)}\xi(p)$ to get $\mathbb{E}_p(N_r) \leq e^{-1}/3$, as claimed. □

The following celebrated result of Smirnov and Werner [27] gives an asymptotic 'power law' behavior for the parameters $\chi(p)$ and $\xi(p)$ of the triangular lattice, as p approaches the critical value $1/2$ from below.

Theorem 6 [27]. *As* p *approaches* $1/2$ *from below, the following asymptotic behavior holds*:

$$\chi(p) = (1/2 - p)^{-43/18 + o(1)} \quad \textit{and} \quad \xi(p) = (1/2 - p)^{-4/3 + o(1)}.$$

Appendix E. A Collection of Suitable PDAGs and Their Colorings for Small Group Size

In this Appendix (Figs. 10, 11, 12), we present the PDAGs with smallest number of edges (e) and vertices (v) we obtained for $t \in \{1, 2, 3\}$. Surprisingly, for those values of t, the graphs with the smallest number of edges were also those with the smallest number of vertices. However, we were not able either to prove or disprove this fact in the general case.

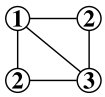


Fig. 10. $n = 3, t = 1, e = 5, v = 4$.

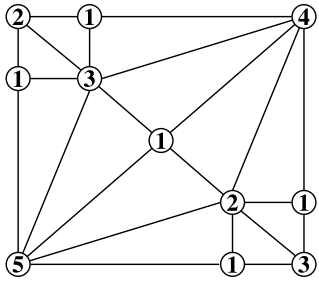


Fig. 11. $n = 5, t = 2, e = 22, v = 11$.

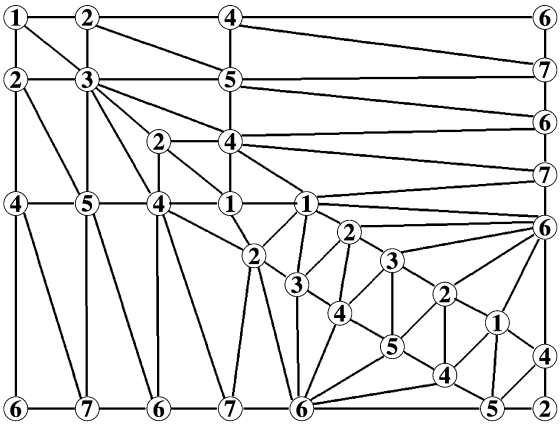


Fig. 12. $n = 7, t = 3, e = 84, v = 35$.

Remark 18. The graph obtained for $t = 1$ is optimal since any PDAG must have at least four vertices to allow paths to exist. It can also be seen that such a graph must have at least five edges.

We also performed some implementations using the square grid $\mathcal{G}_{\text{tri}}(\ell, \ell)$. In the case, $t = 2$ and $\ell = 4$, we obtained 36084 different colorings (each of them generating 120 similar colored grids by permuting over the set of five colors). Out of these 36084 solutions, 89 were symmetric (i.e. the colored grid was its own transposed). We were able to extend these 89 symmetric solutions into 4820 symmetric solutions for the case $t = 3$ and $\ell = 6$ by adding two rows below and two columns on the right hand side of the original ($t = 2, \ell = 4$) symmetric grid.

References

[1] N. Alon, J.H. Spencer, *The Probabilistic Method* (Wiley-Interscience, New York, 2000)

[2] J. Bar-Ilan, D. Beaver, Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction, in *8th Annual ACM Symposium on Principles of Distributed Computing* (ACM, New York, 1989), pp. 201–209

[3] D.A. Barrington, Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1, in *18th Annual ACM Symposium on Theory of Computing* (ACM, New York, 1986), pp. 1–5

[4] M. Ben-Or, S. Goldwasser, A. Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computation, in *20th Annual ACM Symposium on Theory of Computing* (ACM, New York, 1988), pp. 1–10

[5] J. Benaloh, Secret sharing homomorphisms: keeping shares of a secret, in *Advances in Cryptology: Crypto'86*. Lecture Notes in Computer Science, vol. 263 (Springer, Berlin, 1987), pp. 251–260

[6] B. Bollobàs, O. Riordan, *Percolation* (Cambridge University Press, Cambridge, 2006)

[7] D. Chaum, C. Crépeau, I. Damgård, Multi-party unconditionally secure protocols, in *20th Annual ACM Symposium on Theory of Computing* (ACM, New York, 1988), pp. 11–19

[8] R. Cramer, S. Fehr, Y. Ishai, E. Kushilevitz, Efficient multi-party computation over rings, in *Advances in Cryptology: Eurocrypt'03*. Lecture Notes in Computer Science, vol. 2656 (Springer, Berlin, 2003), pp. 596–613

[9] I. Damgård, J.B. Nielsen, Scalable and unconditionally secure multiparty computation, in *Advances in Cryptology—Crypto'07*. Lecture Notes in Computer Science, vol. 4622 (Springer, Berlin, 2007), pp. 572–590

[10] I. Damgård, M. Fitzi, E. Kiltz, J.B. Nielsen, T. Toft, Unconditionally secure constant-rounds multiparty computation for equality, comparison, bits and commitments, in *3rd Theory of Cryptography Conference*. Lecture Notes in Computer Science, vol. 3876 (Springer, Berlin, 2006), pp. 285–304

[11] I. Damgård, Y. Ishai, M. Krøigaard, Perfectly secure multiparty computation and the computational overhead of cryptography, in *Advances in Cryptology—Eurocrypt'10*. Lecture Notes in Computer Science, vol. 6110 (Springer, Berlin, 2010), pp. 445–465

[12] Y. Desmedt, Y. Wang, M. Burmester, A complete characterization of tolerable adversary structures for secure point-to-point transmissions, in *16th International Symposium on Algorithms and Computation*. Lecture Notes in Computer Science, vol. 3827 (Springer, Berlin, 2005), pp. 277–287

[13] Y. Desmedt, J. Pieprzyk, R. Steinfeld, H. Wang, On secure multi-party computation in black-box groups, in *Advances in Cryptology—Crypto'07*. Lecture Notes in Computer Science, vol. 4622 (Springer, Berlin, 2007), pp. 591–612

[14] R. Diestel, *Graph Theory*, 2nd edn. Graduate Texts in Mathematics (Springer, Berlin, 2000)

[15] W. Diffie, M.E. Hellman, New directions in cryptography. *IEEE Trans. Inf. Theory* **22**(6), 644–654 (1976)

[16] T. El Gamal, A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory* **31**(4), 469–472 (1985)

[17] Y. Frankel, Y. Desmedt, M. Burmester, Non-existence of homomorphic general sharing schemes for some key spaces (extended abstract), in *Advances in Cryptology—Crypto'92*. Lecture Notes in Computer Science, vol. 740 (Springer, Berlin, 1993), pp. 549–557

[18] O. Goldreich, *Foundations of Cryptography: Basic Applications*, vol. II (Cambridge University Press, Cambridge, 2004)

[19] J.M. Hammersley, Percolation processes: lower bounds for the critical probability. *Ann. Math. Stat.* **28**(3), 790–795 (1957)

[20] M. Hirt, U. Maurer, Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract), in *16th Annual ACM Symposium on Principles of Distributed Computing* (ACM, New York, 1997), pp. 25–34

[21] H. Kesten, *Percolation Theory for Mathematicians* (Birkhäuser, Basel, 1982)

[22] S.S. Magliveras, D.R. Stinson, T. van Trung, New approaches to designing public key cryptosystems using one-way functions and trapdoors in finite groups. *J. Cryptol.* **15**(4), 285–297 (2002)

[23] A. Myasnikov, V. Shpilrain, A. Ushakov, *Group-Based Cryptography*. Advanced Courses in Mathematics—CRM Barcelona (Birkhäuser, Basel, 2008)

[24] S.-H. Paeng, K.-C. Ha, J.H. Kim, S. Chee, C. Park, New public key cryptosystem using finite non Abelian groups, in *Advances in Cryptology—Crypto'01*. Lecture Notes in Computer Science, vol. 2139 (Springer, Berlin, 2001), pp. 470–485

[25] R.L. Rivest, A. Shamir, L.M. Adleman, A method for obtaining digital signatures and public key cryptosystems. *Commun. ACM* **21**(2), 120–126 (1978)

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

[27] S. Smirnov, W. Werner, Critical exponents for two-dimensional percolation. *Math. Res. Lett.* **8**, 729–744 (2001)

[28] X. Sun, A.C.-C. Yao, C. Tartary, Graph design for secure multiparty computation over non-Abelian groups, in *Advances in Cryptology—Asiacrypt'08*. Lecture Notes in Computer Science, vol. 5350 (Springer, Berlin, 2008), pp. 37–53

[29] N.R. Wagner, M.R. Magyarik, A public key encryption scheme based on the word problem, in *Advances in Cryptology—Crypto'84*. Lecture Notes in Computer Science, vol. 196 (Springer, Berlin, 1985), pp. 19–36