Active Security in Multiparty Computation over Black-Box Groups

Yvo Desmedt[1], Josef Pieprzyk[2], and Ron Steinfeld[3]*

[1] Dept. of Computer Science, University College London, UK
y.desmedt@cs.ucl.ac.uk
[2] Centre for Advanced Computing – Algorithms and Cryptography (ACAC)
Dept. of Computing, Macquarie University, North Ryde, Australia
josef.pieprzyk@mq.edu.au
[3] Clayton School of Information Technology,
Monash University, Clayton, Australia
ron.steinfeld@monash.edu

Abstract. Most previous work on unconditionally secure multiparty computation has focused on computing over a finite field (or ring). Multiparty computation over other algebraic structures has not received much attention, but is an interesting topic whose study may provide new and improved tools for certain applications. At CRYPTO 2007, Desmedt et al introduced a construction for a passive-secure multiparty multiplication protocol for black-box groups, reducing it to a certain graph coloring problem, leaving as an open problem to achieve security against *active* attacks.
We present the first n-party protocol for unconditionally secure multiparty computation over a black-box group which is secure under an *active* attack model, tolerating any adversary structure Δ satisfying the Q^3 property (in which no union of three subsets from Δ covers the whole player set), which is known to be necessary for achieving security in the active setting. Our protocol uses Maurer's Verifiable Secret Sharing (VSS) but preserves the essential simplicity of the graph-based approach of Desmedt et al, which avoids each shareholder having to rerun the full VSS protocol after each local computation. A corollary of our result is a new active-secure protocol for general multiparty computation of an arbitrary Boolean circuit.

Key Words: Multi-Party Computation, General Adversary Structures, Non-Abelian Group, Black-Box, Graph Colouring, Active Security.

1 Introduction

Multiparty computation in the unconditionally secure model has been extensively studied in the cryptographic literature. The classical works [3, 4] established secure protocols against threshold adversary structures (with the number

* Most of this work was done while R.S. was with Macquarie University.

of corrupted parties $t < n/3$, which was shown optimal), and later protocols with improved efficiency and security against general adversary structures were presented [15, 10, 11, 6, 16, 18, 17, 19, 2, 7]. Yet a common feature of those protocols is that they reduce general multiparty computation to performing addition and multiplication computations over a *field*. This raises the natural problem of realizing multiparty computation over other algebraic structures. The question is interesting not only intrinsically from a theoretical point of view, but may lead to new techniques that may have advantages over those used for multiparty computation over fields. Generalizations of the field-based protocols to work over a *ring* have been investigated (e.g. [5]), but the problem of performing multiparty computation over an arbitrary *group* has received less attention so far, with only passive-secure protocols known [9]. The problem of multiparty computation over a *non-abelian* group is of particular interest, since, as pointed out in [8], a result due to Barrington [1] (see Sec. 3.4) implies that multiparty computation over the non-abelian symmetric group S_5 is *complete* for general multiparty computation, i.e. it allows construction of secure computation protocols for *arbitrary* functions.

Let G denote a finite (multiplicatively written) group, and C denote a G-circuit, i.e. a circuit in which the inputs are elements of G and the allowed circuit gates are either a multiplication gate (taking two inputs in G and outputting their product in G) or a constant multiplication gate (taking an input in G and returning the input multiplied by some constant in G). A multiparty computation protocol for C over a black box group G [8] is an n-party protocol in which, for $i = 1, \dots, m$, input $x_i \in G$ is held by one of the n parties, and at the end of the protocol, all parties hold $y = f_C(x_1, \dots, x_m) \in G$, where f_C is the function over G computed by the G-circuit C. The protocol is said to be *black-box* if it treats the group G as a black-box: the only operations performed in the protocol are sampling random elements in G, multiplying elements in G and computing inverses in G.

Desmedt et al [9, 8] introduced a novel construction for a black-box protocol for f_C, by reducing it to a certain graph coloring problem. The approach of [9] differs in an interesting way from classical multiparty computation protocols that work over fields and rings [3, 4]. The latter protocols designed to handle an adversary structure Δ (specifying the collection of party subsets that may be corrupted) make use of a secret sharing scheme SS_Δ secure against adversary structure Δ; to multiply two circuit values shared among the parties using SS_Δ, each party does a local multiplication operation on its shares, and then performs a full SS_Δ sharing to reshare the result among *all* parties, who then perform a recombination operation to compute their new share. In contrast, in the group-based protocol of [9], two circuit values shared by a secret sharing scheme SS_Δ are multiplied by a sequence of *simple* resharing and combining operations specified by a colored communication graph; each of these sharing operations are typically much simpler that running a full resharing operation for SS_Δ (for instance, in the scheme of [9], a sharing operation just involves computing a 2-of-2 sharing of a group element and sending the two shares to two parties).

Unfortunately, the protocol of [9] only achieves security against passive adversaries, since it does not provide a way of verifying the correctness of the computations performed. It thus left the natural open problem of designing multiparty protocols over a black-box group secure against *active* adversaries.

Our Results. In this paper, we address the above-mentioned open problem. We present the first n-party protocol for unconditionally secure multiparty computation over a black-box group G which is secure under an *active* attack model. Our protocol achieves the *optimal* resilience in the active setting, namely it is can tolerate any adversary structure Δ satisfying the Q^3 property, in which no union of three subsets from Δ covers the whole player set, which is known to be necessary for achieving security in the active setting [15]. The communication complexity of our protocol for computing a G-circuit C is $O(|C| \cdot M(\Delta)^2 \cdot \text{poly}(n))$ group elements, where $M(\Delta)$ denotes the number of maximal sets in the adversary structure Δ, and $|C|$ denotes the size of C.

A corollary of our result is a new active-secure protocol for securely computing an arbitrary Boolean circuit C, via Barrington's result mentioned above, with communication complexity $O(|C| \cdot M(\Delta)^2 \cdot \text{poly}(n))$ bits. Note that a similar communication complexity proportional to $M(\Delta)^2$ is achieved in [20] but using a completely different field-based approach.

Our construction is based on an extension of the communication graph approach used in [9]. However, whereas in [9] each node in the graph is assigned a single color corresponding to the party that performs a multiplication and resharing computation at that node, in our protocol each edge is assigned a *subset* of colors corresponding to a subset of players that send or receive shares along the edge and jointly participate in the computation performed at the graph nodes adjacent to the edge. To ensure the validity of the initial input sharing, we use Maurer's simple construction of a Verifiable Secret Sharing (VSS) scheme [19], which works over a black-box group. At each internal node of our graph, the correctness of the computation is verified by a multiparty pairwise comparison protocol inspired by Maurer's field-based multiplication protocol from [19].

Interestingly, unlike Maurer's field-based multiplication protocol in [19], our protocol retains the essential simplicity of the resharing operations used at each node of the graph in the protocol of [9]: each party in the subset of players assigned to a node in our protocol does *not* rerun the VSS sharing protocol for resharing the intermediate protocol multiplication values at each node, but uses just a 2-of-2 sharing of its output value, and consequently has an efficiency advantage over Maurer's protocol when applied to computing Boolean circuits (see Sec. 3.4 for more details). Please note however, that similarly to Maurer's protocol in [19], we do not claim that our Boolean circuit protocol offers any asymptotic complexity advantages over previous field-based protocols for Boolean circuit computation. Indeed, the field-based protocol from [20] offers a similar asymptotic complexity for general adversary structures, and the protocols from [3, 4] are asymptotically significantly more efficient for threshold structures. Rather, we view it mainly as an illustration of the power of our protocols over black-box groups.

Due to limited space, the proofs of some results have been omitted from this version of the paper. They can be found the full version, available from the authors' web page.

Open Problems. A central and interesting open problem left by our work is to construct an active-secure protocol over black box groups tolerating a more restricted but useful class of adversary structures, such as a t-of-n threshold adversary structure, while achieving a communication complexity *polynomial* in n. Currently, we do not see how to adapt our approach to achieve this goal, and it seems to require new ideas. In particular, there seems to be an inherent contradiction between the requirement to have at least $2t+1$ colors assigned to each edge in our protocol graph (in order to achieve an honest majority at the node and thus ensure correctness of the computation at the node) and the optimal resilience condition $n = 3t + 1$, which means that each edge excludes a *unique* t-subset of parties. The security of our approach (like that of [9]) against a t-subset I of parties requires the existence of an I-avoiding path in the graph whose edges exclude I. Thus if each edge excludes only a unique t-subset, that edge can be used for only one I-avoiding path, whereas in a polynomial-sized graph, each edge must be re-used for exponentially many paths, since there are exponentially many t-subsets I.

Other Related Work. Sun et al [21] gave improvements to the graph coloring constructions of Desmedt et al [9], showing them to be polynomial-sized for certain resilience cases. These apply to the 't-reliable' coloring notion needed for the passive-secure protocols in [9], but do not seem applicable to our stronger 'Δ-active-reliable' coloring notion that we use to achieve active-security, and involves coloring the graph edges with $2t{+}1$-subsets of colors. As discussed above, this notion seems to require exponential-sized graphs. Barrington's encoding of Boolean circuits into S_5 was used in secure multiparty computation already in 1987 [14]. However, the security achieved in [14] was in the computationally bounded attack model, while in [8] and in this paper, the security achieved is against computationally unbounded attacks.

2 Preliminaries

2.1 Active Attack Model

We first recall the formal definition of secure multi-party computation in the active (malicious), computationally unbounded attack model, restricted to deterministic symmetric functionalities and perfect emulation [13]. The number of parties participating in the protocol is denoted by n, and the parties are denoted by $P_1, \ldots, P_n$. We assume a general static party corruption model specified an *adversary structure* Δ, which is a (monotone) collection of subsets of the player index set $[n] = \{1, \ldots, n\}$, corresponding to the player subsets that may be corrupted. It is known [15] that secure multiparty computation in the active computationally unbounded model is possible for an adversary structure Δ if and only if Δ has the Q^3 property, i.e. $[n] \neq I_1 \cup I_2 \cup I_3$ for all $I_1, I_2, I_3 \in \Delta$.

The uncorrupted players are assumed to correctly follow the protocol, whereas the corrupted players can behave arbitrarily and are allowed to communicate with each other. Also, every pair of parties can communicate via a private authenticated channel (meaning that communication between two parties P_i and P_j cannot be eavesdropped by any other party, and that when P_i receives a message from P_j, P_i knows that the message was sent by P_j; moreover, an honest player P_i can detect that an expected message from P_j has not arrived - in this case, we assume that P_i substitutes a certain 'default' message, as specified in the protocol).

Security in the active model must guarantee not only the *privacy* of the inputs held by the honest parties (as in the passive case), but also the *correctness* of the protocol output computed by the honest parties. But note that perfect correctness can never be achieved in the following sense: regardless of the protocol, nothing can prevent an adversary-controlled party P_i from ignoring its protocol input x_i and *substituting* a different value x'_i when participating in the protocol. Accordingly, the security definition is constructed to ensure that this *substitution* "attack" is essentially the only attack possible on correctness of the protocol (in particular, it ensures that the substituted value x'_i cannot depend on the values of honest party inputs). To achieve this, the security definition compares the execution of the real protocol in question (called the REAL model), to the execution of an idealized protocol involving an honest trusted entity in addition to the parties running the protocol (called the IDEAL model). In the IDEAL model, each party privately sends its (possibly substituted) protocol input to the honest trusted entity. The trusted entity evaluates the desired function f on the inputs it received and sends the result back to all parties. It is clear that in the IDEAL model, the 'substitution' attack is the only possible attack. So, a protocol is said to be secure if for every adversary $\mathcal{A}$ in the REAL model, there is an adversary B in the IDEAL model which produces the same output distribution (for the honest parties and the adversary). We now present the formal definition.

Definition 1. *Let $f : (\{0,1\}^*)^n \rightarrow \{0,1\}^*$ 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 $\boldsymbol{x} = (x_1, \ldots, x_n)$, and the projection of the n-ary sequence $\boldsymbol{x}$ on the coordinates in $I \subseteq [n]$ by $\boldsymbol{x}_I$. Let $\mathcal{A}$ denote a REAL model adversary against protocol $\prod$, where $\mathcal{A}$ controls a subset $I \in \Delta$ of corrupted parties. Let $\mathsf{OUT}^{\prod}_{I,A}(\boldsymbol{x})$ (respectively $\mathsf{OUT}^{\prod}_{[n]\setminus I,A}(\boldsymbol{x})$) denote the vector of outputs of the corrupted players P_i with $i \in I$ using some standard ordering (respectively, the list of outputs of honest players P_i with $i \in [n] \setminus I$ using some standard ordering) after running protocol $\prod$ on input $\boldsymbol{x}$, with $\mathcal{A}$ run on input $(\boldsymbol{x}_I, I)$ and controlling parties P_i for $i \in I$.*

*We say that $\prod$ is a Δ-**secure protocol for computing** f if, for every REAL model adversary $\mathcal{A}$, there exists an IDEAL model adversary $B = (B_1, B_2)$ such* *that, for all $I \in \Delta$ and for all $\boldsymbol{x} \in (\{0,1\}^*)^n$, the random variables $\mathsf{REAL}^{\Pi}_{I,\mathcal{A}}(\boldsymbol{x})$ and $\mathsf{IDEAL}^{f}_{I,B}(\boldsymbol{x})$ are identically distributed, where we define:*

$$\mathsf{REAL}^{\Pi}_{I,\mathcal{A}}(\boldsymbol{x}) = (\mathsf{OUT}^{\Pi}_{[n]\setminus I,A}(\boldsymbol{x}), \mathsf{OUT}^{\Pi}_{I,A}(\boldsymbol{x}))$$

and

$$\mathsf{IDEAL}^{f}_{I,B}(\boldsymbol{x}) = (f(\boldsymbol{x'})^{n-t}, B_2(\boldsymbol{x}_I, I, f(\boldsymbol{x'}); r))$$

with $\boldsymbol{x'} = (x'_1, \ldots, x'_n)$, $x'_i = B_1(\boldsymbol{x}_I, I, i; r)$ for $i \in I$ and $x'_i = x_i$ for $i \in [n] \setminus I$. Here, r is the (common) uniformly random coins input of deterministic algorithms B_1 and B_2.

Note that in the IDEAL model adversary $B = (B_1, B_2)$ in the above definition, algorithm B_1 performs the substitution of corrupted player inputs, whereas B_2 simulates the output of the corrupted players. The first component of $\mathsf{IDEAL}_{I,B}(\boldsymbol{x})$, namely $f(\boldsymbol{x'})^{n-t}$ represents the $n-t$ outputs of the honest players indexed by $[n] \setminus I$ in the IDEAL model; these outputs are all equal to $f(\boldsymbol{x'})$.

2.2 Maurer's Simple Verifiable Secret Sharing Scheme

Our protocol makes use of a Verifiable Secret Sharing (VSS) scheme due to Maurer [19], which works over any black-box group. We now recall this scheme.

First, we recall the definition of VSS. It is an adaptation of standard secret sharing to the active-security setting, in which both the dealer and some shareholders may be actively corrupted.

Definition 2 (VSS). *A VSS scheme is run among n parties $P_1, \ldots, P_n$, one of which is the* dealer. *The players with indices in $I \in \Delta$ (possibly including the dealer) are actively corrupted, and all other players honestly follow the protocol. It consists of two protocols: a sharing protocol **VSS Share** used by the dealer to distribute shares of his secret among all parties, and a **VSS Reconstruct** protocol used by the shareholders to reconstruct the secret. The protocols satisfy the following conditions:*

- ***Unique Reconstruction:** At the end of a run of the **VSS Share** protocol, the dealer is committed to a unique secret s, in the following sense: a subsequent run of **VSS Reconstruct** ends with all honest parties returning the value s.*
- ***Honest Dealer Correctness:** If the dealer is honest with secret s, **VSS Reconstruct** ends with all honest parties returning the value s.*
- ***Honest Dealer Privacy:** If the dealer is honest, the distribution of the adversary's view during the **VSS Share** protocol is independent of the dealer's secret s.*

Let us now recall Maurer's simple VSS scheme [19] for adversary structure Δ. We denote by $M(\Delta)$ the number of *maximal* sets in Δ. Below, G denotes any

Protocol 1 Maurer's **VSS Share**

```
Input: Dealer holds a secret $s \in G$.
1: Let $\ell = M(\Delta)$ and $I_1, \ldots, I_\ell$ denote the sequence of all maximal sets in $\Delta$
   (in some ordering).
2: Dealer chooses uniformly random shares $s_1, \ldots, s_\ell$ in $G$ such that
   $s_1 s_2 \cdots s_\ell = s$.
3: For $i \in [\ell]$, dealer sends $s_i$ to each party $P_j$ with $j \in [n] \setminus I_i$.
4: For $i \in [\ell]$, every pair of parties $P_j, P_k$ with $j, k \in [n] \setminus I_i$ check (by exchanging
   values) whether their received values of $s_i$ agree. If any party detects a
   disagreement, it broadcasts a complaint.
5: Dealer broadcasts to all parties all shares $s_i$ for which a complaint was broad-
   cast.
Output: Party $P_j$ holds shares $\{s_i : j \in [n] \setminus I_i\}$, for $j \in [n]$.
```

Protocol 2 Maurer's **VSS Reconstruct**

```
Input: Party $P_j$ holds shares $\{s_i : j \in [n] \setminus I_i\}$, for $j \in [n]$.
1: For $j \in [n]$, party $P_j$ sends its shares $\{s_i : j \in [n] \setminus I_i\}$, to every other party.
2: For $i \in [\ell]$, each party $P$ reconstructs $s_i$ as the unique value $v$ for which
   there exists a $J \in \Delta$ such that $P$ received $v$ as the value of $s_i$ in the previous
   step from all parties $P_j$ with $j \in [n] \setminus (I_i \cup J)$.
3: Each party reconstructs $s = s_1 \cdots s_\ell$, where for $i \in [\ell]$, $s_i$ is the value recon-
   structed in the previous step.
Output: Party $P_j$ holds secret $s$, for $j \in [n]$.
```

(black-box) group. Also note that, if Δ is Q^3, a broadcast from one player to all players (as used below) can be simulated with communication polynomial in n using only point to point communication links between any pair of players [12].

Theorem 1 ([19]). *If Δ is Q^3, then Maurer's **VSS Share** and **VSS Reconstruct** protocols form a VSS scheme (i.e. secure against adversary structure Δ). The communication complexity of **VSS Share** and **VSS Reconstruct** is $O(M(\Delta) \cdot poly(n))$ group elements.*

2.3 G-Circuits

We recall the definition of G-circuits. 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 $x \cdot y \in G$[4]
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).

[4] The incoming edges to Mult gates need to be labeled to indicate which one is the left input.

We denote by $f_C : G^m \to G$ the function computed by the G-circuit C.

3 Our New Protocol

First, in Sec. 3.1, we reduce the G-circuit computation protocol problem (in which at the beginning, each party holds $x_i \in G$, and at the end each party holds the circuit output) to the Shared 2-Product protocol problem (in which at the beginning, the parties hold shares of two elements $x, y \in G$, and at the end, the parties holds shares of $z = x \cdot y$). This part of our protocol is almost identical to that of [9]. Then, in Sec. 3.2, we show how to construct a Shared 2-Product protocol, using a suitable coloring of a certain planar graph. In this part we introduce significant modifications to the protocol in [9] to handle active attacks.

3.1 Construction of G-Circuit Protocol from a Shared 2-Product Subprotocol

We begin by reducing the problem of constructing a Δ-private protocol for computing an m-input G-circuit computing a function $f_C(x_1, \ldots, x_m)$ (where each input x_i is held by one of the parties), 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 this subprotocol *active* correctness and strong Δ-security properties, which strengthen the correctness and strong Δ-privacy of the passive model in [9] to the active case. In the definition below, the share ownership functions $\mathcal{O}_x, \mathcal{O}_y, \mathcal{O}_z$ specify for each share index $j \in [\ell]$, the indices of the sets of players $\mathcal{O}_x(j), \mathcal{O}_y(j), \mathcal{O}_z(j) \subseteq [n]$ which hold the jth input shares $s_x(j)$ and $s_y(j)$ and jth output share $s_z(j)$, respectively.

Definition 3 (Shared n-Party 2-Product Subprotocol). *A n-Party Shared 2-Product subprotocol $\prod_S$ with sharing parameter ℓ and share ownership functions $\mathcal{O}_x, \mathcal{O}_y, \mathcal{O}_z : [\ell] \to 2^{[n]}$ has the following features:*

- *Input: For $j \in [\ell]$, each party in set $\mathcal{O}_x(j)$ holds jth share $s_x(j) \in G$ of x, and each party in set $\mathcal{O}_y(j)$ holds jth share $s_y(j) \in G$ of y, where $x \overset{\text{def}}{=} s_x(1) \cdot s_x(2) \cdots s_x(\ell)$ and $y \overset{\text{def}}{=} s_y(1) \cdot s_y(2) \cdots s_y(\ell)$, respectively.*
- *Output: For $j \in [\ell]$, each party in set $\mathcal{O}_z(j)$ holds jth share $s_z(j)$ of output $z \overset{\text{def}}{=} s_z(1) \cdots s_z(\ell)$.*
- *Active-Correctness: We say that $\prod_S$ is **active-correct** if it has the following property. Suppose that, at the beginning of the protocol, for $j \in [\ell]$, all honest parties in set $\mathcal{O}_x(j)$ (resp. $\mathcal{O}_y(j)$) hold the same input share $s_x(j)$ (resp. $s_y(j)$), defining protocol inputs $x = s_x(1) \cdots s_x(\ell)$ and $y = s_y(1) \cdots s_y(\ell)$. Then, at the end of the protocol, for each $j \in [\ell]$, all honest parties in set $\mathcal{O}_z(j)$ hold the same output share $s_z(j)$ defining protocol output $z = s_z(1) \cdots s_z(\ell)$, and $z = x \cdot y$ holds.*

– *Strong Δ-Security: Let $\mathcal{A}$ denote a REAL model adversary against subprotocol $\prod_S$, where $\mathcal{A}$ controls a subset $I \in \Delta$ of corrupted parties. Let $I_x = \{j \in [\ell] : (\mathcal{O}_x(j) \cap I) \neq \emptyset\}$ and $I_y = \{j \in [\ell] : (\mathcal{O}_y(j) \cap I) \neq \emptyset\}$. Let $\mathcal{A}^{\prod_S(s_x,s_y)}(\{s_x(j)\}_{j \in I_x}, \{s_y(j)\}_{j \in I_y}, z_{aux})$ denote the output state of $\mathcal{A}$ at the end of a run of subprotocol $\prod_S$ with protocol inputs s_x, s_y, in which $\mathcal{A}$ is run on input $(\{s_x(j)\}_{j \in I_x}, \{s_y(j)\}_{j \in I_y}, z_{aux})$, where z_{aux} is an auxiliary input (representing the adversary's input state), and let $s_z(j)$ denote the jth output share held by the honest parties in set $\mathcal{O}_z(j)$ at the end of this run. We say that $\prod_S$ achieves **strong Δ-security** if, for every $I \in \Delta$, there exist $j_x^*, j_y^*, j_z^* \in [\ell]$ with $j_z^* \in \{j_x^*, j_y^*\}$ and sets $\mathcal{O}_x(j_x^*), \mathcal{O}_y(j_y^*), \mathcal{O}_z(j_z^*)$ all disjoint from I, such that for every active adversary $\mathcal{A}$ against $\prod_S$ corrupting parties P_i for $i \in I$, there exists a probabilistic simulator algorithm S such that for all protocol inputs $\boldsymbol{s}_x, \boldsymbol{s}_y$ and auxiliary inputs z_{aux}, the random variables $\mathsf{REAL}_{I,\mathcal{A}}^{\prod_S}$ and $\mathsf{SIM}_{I,\mathsf{S}}^{\prod_S}$ are identically distributed. Here, we define:*

$$\begin{aligned}
\mathsf{REAL}_{I,\mathcal{A}}^{\prod_S} &= \langle \mathcal{A}^{\prod_S(\boldsymbol{s}_x,\boldsymbol{s}_y)}(\{s_x(j)\}_{j \in I_x}, \{s_y(j)\}_{j \in I_y}, z_{aux}), \{s_z(j)\}_{j \in [\ell] \setminus \{j_z^*\}} \rangle, \\
\mathsf{SIM}_{I,\mathsf{S}}^{\prod_S} &= \mathsf{S}(I, \{\mathbf{s}_x(j)\}_{j \in [\ell] \setminus \{j_x^*\}}, \{\mathbf{s}_y(j)\}_{j \in [\ell] \setminus \{j_y^*\}}, z_{aux}).
\end{aligned}$$

*If $j_z^* = j_x^*$ (resp. $j_z^* = j_y^*$) then we say $\prod_S$ achieves **x-preserving strong Δ-security** (resp. **y-preserving strong Δ-security**). If $j_z^* = j_x^* = j_y^*$ for all I, then we say $\prod_S$ achieves **symmetric strong Δ-security**.*

Remark. In the above definition, the simulator S must simulate both the output of $\mathcal{A}$ and all but one of the output shares $s_z(j)$, given all but one of the x-input (resp. y-input) shares $s_x(j)$ (resp. $s_y(j)$).

Our construction of an active-secure G-circuit computation protocol $\prod_a(C, \prod_S)$ given a G-circuit C with m input nodes, and a Shared 2-Product subprotocol $\prod_S$, runs similarly to the corresponding passive construction in [9], except that here, the secrets x_i are shared out using Maurer's VSS scheme, and each share is held by a set of parties, rather than a single party. Due to space limitations, we defer the formal specification of protocol $\prod_a(C, \prod_S)$ to the full version of the paper. We assume that $\prod_S$ satisfies symmetric strong Δ-security, with sharing parameter ℓ and share ownership functions $\mathcal{O}_x = \mathcal{O}_y = \mathcal{O}_z$ (for simplicity, we do not consider here the more general case of x-preserving or y-preserving strong Δ-security as in [9] since our constructions for Π_S in later sections satisfy symmetric strong Δ-security). Since our protocol makes use of Maurer's VSS scheme (see Sec. 2.2), we also assume here for compatibility that the sharing parameter $\ell = M(\Delta)$, and that $\mathcal{O}_x(i) = [n] \setminus I_i$, for $i \in [\ell]$, where I_i is the ith t-subset of $[n]$ (in some ordering), as used in Maurer's VSS scheme. Below, for $i \in [m]$, we let $j(i) \in [n]$ denote the index of the party holding the ith circuit input x_i.

The following lemma establishes the Δ-security of protocol $\prod_a(T, \prod_S)$, assuming the active-correctness and strong Δ-security of subprotocol $\prod_S$. The IDEAL model adversary in the proof makes use of the unique reconstruction property of the VSS scheme to reconstruct from the shares held by the honest parties, the 'substituted' input values x'_i committed by the corrupted players during the dealing phase of the VSS. The IDEAL model adversary then simulates the view of the corrupted parties at each node of the tree T by using the known inputs to the subprotocol run at the node as input to the simulator associated to subprotocol Π_S thanks to its strong Δ-security. This lemma can be viewed as an extension of Lemma 3 in [9] to the active attack setting. Its proof can be found in the full version.

Lemma 1. *For any G-circuit C, if the n-party Shared 2-Product subprotocol $\prod_S$ satisfies active-correctness and symmetric strong Δ-security, then the protocol $\prod_a(C, \prod_S)$ is an n-party Δ-secure protocol for computing function f_C computed by C.*

3.2 Construction of a t-Secure Shared 2-Product Subprotocol from a t-Active-Reliable Coloring

We now show how to reduce the problem of constructing a t-Private n-Party Shared 2-Product Subprotocol $\prod_S$ to a certain combinatorial coloring problem for a planar graph. In contrast to the coloring in [9] in which graph *nodes* are assigned colors, our coloring assigns colors to graph *edges*. More significantly, whereas in [9] each node was assigned a *single color* from $[n]$ denoting the index of the party performing computation at that node, we assign a *subset* of colors from $[n]$ to each edge, denoting the indices of parties receiving the share sent along the edge, and participating in the computation at the node that the edge is directed towards. Our construction is specific to the PDAG $\mathcal{G}_{grid}(\ell)$ shown in Fig. 1, with $\ell = M(\Delta)$, the number of maximal sets in the adversary structure Δ. The node rows (resp. columns) of $\mathcal{G}_{grid}(\ell)$ are numbered consecutively from

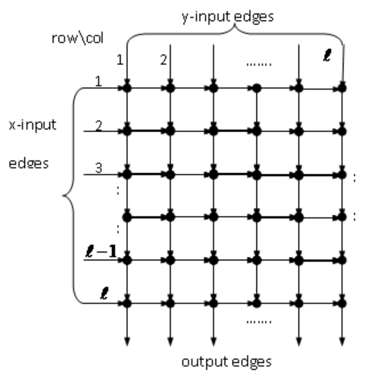

Fig. 1. The PDAG $\mathcal{G}_{grid}(\ell)$.

1 to ℓ from top to bottom (resp. left to right). We label the edges of $\mathcal{G}_{grid}(\ell)$ as follows: the label (i, j, d) denotes the edge of $\mathcal{G}_{grid}(\ell)$ which is directed into the node in the ith row and jth column in the direction $d \in \{H, V\}$ (horizontal if $d = H$ or vertical if $d = V$). An exception is that $(\ell + 1, j, V)$ denotes the jth

outgoing edge of the node in row ℓ and column j. Note also that the nodes on column ℓ do not have horizontal outgoing edges. We call the horizontal incoming edges to the leftmost column the *x-input edges* (and edge $(\ell+1-j, 1, H)$ is called the jth x-input edge), the vertical incoming edges to the top row the *y-input edges* (and edge $(1, j, V)$ is called the jth y-input edge), and the vertical outgoing edges in the bottom row the *output edges* (and edge $(\ell+1, j, V)$ is called the jth output edge).

Let $C : [\ell+1] \times [\ell] \times \{H, V\} \to 2^{[n]}$ be an n-Active-Reliable coloring function that associates to each edge (i, j, d) a color subset $C(i, j, d)$ from the set of n possible colors $[n]$. We now define the notion of a Δ-active-reliable n-coloring, which may be viewed as an 'active' variant of the t-reliable coloring in [9].

Definition 4 (Δ-active-reliable n-coloring). *We say that $C : [\ell+1] \times [\ell] \times \{H, V\} \to 2^{[n]}$ is a Δ-active-reliable n-coloring for PDAG $\mathcal{G}_{grid}(\ell)$ if $C(i, j, H) \cap C(i, j, V) \notin \Delta$ and $C(i, j, d) \neq I_1 \cup I_2$ for all i, j, d and $I_1, I_2 \in \Delta$, and, for each $I \in \Delta$, there exists $j^* \in [\ell]$ and:*

- *A path $PATH_x$ in $\mathcal{G}_{grid}(\ell)$ from the j^*th x-input edge (i.e. edge $(\ell+1-j^*, 1, H)$) to the j^*th output edge (i.e. edge $(\ell+1, j^*, V)$), such that all edges (i, j, d) along the path have color sets $C(i, j, d)$ disjoint from the subset I (we call such a path I-avoiding), and*
- *An I-avoiding path $PATH_y$ in $\mathcal{G}_{grid}(\ell)$ from the j^*th y-input edge (i.e. edge $(1, j^*, V)$) and the j^*th output edge (i.e. edge $(\ell+1, j^*, V)$).*

If the jth x-input, y-input and output edges are assigned the same color subset by C for all $j \in [\ell]$ (i.e. $C(1, j, V) = C(\ell+1-j, 1, H) = C(\ell+1, j, V)$ for $j \in [\ell]$), then we say that C is a symmetric *Δ-active-reliable n-coloring.*

Given a Δ-active-reliable coloring for PDAG $\mathcal{G}_{grid}$, our Shared 2-Product protocol $\Pi_S(\mathcal{G}_{grid}, C)$ is given below as Protocol 3.

Our protocol makes use of a subprotocol **NodeMult** that is run at each node of the graph $\mathcal{G}_{grid}$ and given below as Protocol 4. At each protocol step, if a party P_i expects to receive a group element a from some other party P_j, and P_i does not receive the group element (because P_j is corrupted and sends nothing), we assume that P_i substitutes the default value 1 for the element a.

Theorem 2. *If C is a symmetric Δ-active-reliable n-coloring for $\mathcal{G}_{grid}(\ell)$ then Shared 2-Product protocol $\prod_S(\mathcal{G}_{grid}(\ell), C)$ achieves active-correctness and strong Δ-security.*

The proof of Theorem 2 is based on the properties of the **NodeMult** protocol stated in Lemma 2 below. Due to limited space, the proofs of Lemma 2 and Theorem 2 are deferred to the full version of this paper. Here, we provide an informal overview of the protocol and its security analysis.

Informal overview of protocol $\Pi_S(\mathcal{G}_{grid}, C)$. The edges of $\mathcal{G}_{grid}$ are labeled with shares sent in the protocol, the nodes of $\mathcal{G}_{grid}$ represent multiplication operations on the shares labelling the incoming edges to the node, and the node product is then reshared along the node outgoing edges. The color subsets assigned by coloring C indicate the indices of players receiving the shares sent along the edge, and the multiplication and resharing computations at each node are performed by the subprotocol **NodeMult** among these parties.

Given a simulatable **NodeMult** subprotocol that produces at each node's outgoing edges a fresh resharing of the product of the incoming edge shares, the Δ-security of $\Pi_S(\mathcal{G}_{grid}, C)$ follows from the existence of adversary-avoiding input-output paths in PDAG $\mathcal{G}_{grid}$ (these paths are guaranteed to exist by the Δ-active-reliable property of coloring C). Thanks to the fresh resharing at each node, the output shares sent along outgoing edges not on the adversary-avoiding paths can be simulated by independent random elements.

The main novelty in our protocol versus the passive-secure protocol in [9] is in the design of the **NodeMult** subprotocol. The fact that each edge share is held by a set of parties containing a sufficiently large subset of honest parties, allows us to design appropriate correctness verification checks in **NodeMult** (reminiscent of those in Maurer's robust multiplication protocol [19] over a field) that ensure the correctness of the computation at each node (whereas in the protocol in [9], each node computation is performed by a single party and may fail if the corresponding party is actively corrupted). An interesting aspect of our protocol is that **NodeMult** verification checks can ensure the correctness of the computation *without* having to rerun the full VSS resharing protocol at each node (only a simple 2-of-2 resharing is needed), whereas in Maurer's multiplication protocol, each pairwise product of shares has to be reshared with a VSS, leading to a lower efficiency. The **NodeMult** protocol is run at each internal node of the graph $\mathcal{G}_{grid}(\ell)$. Before the protocol is run, the parties in the set S (labeling the horizontal incoming edge to the node) each hold a share $s \in G$ and the parties in the set T (labeling the vertical incoming edge to the node) each hold a share $t \in G$. The purpose of the protocol is to compute $s \cdot t$ and reshare this product as $a \cdot b$ where a and b are fresh shares. Accordingly, at the end of the protocol, each of the parties in the set A (labeling the outgoing vertical edge) all hold the share $a \in G$ and each of the parties in the set B (labeling the outgoing horizontal edge) all hold the share $b \in G$, such that $a \cdot b = s \cdot t$. Note that in the coloring construction presented in the next section, we have $A = T$ and $B = S$. The protocol runs in two phases.

In the first phase (lines 1 to 11), each party P_k that holds *both* incoming shares s and t (i.e. each party P_k in $S \cap T$) computes $s \cdot t$ and a fresh resharing of this value (a_k, b_k) with $a_k \cdot b_k = s \cdot t$ (note that by construction of S and T from the Δ-active-reliable coloring it is guaranteed that $S \cap T \notin \Delta$ so $S \cap T$ contains at least one *honest* party). Each P_k privately sends its share a_k (resp. b_k) to each party in A (resp. B), and then the parties in A (resp. B) check by doing pairwise comparisons that they all hold the same value of a_k (resp. b_k) for all k. If an inconsistency is detected, the value of a_k (resp. b_k) is broadcast by P_k to all parties. This doesn't violate privacy because it only happens when some party who received or sent a_k (resp. b_k) was corrupted. In the second phase (lines 12 to 26), the parties in A and B check that the sharings (a_k, b_k) define the same secret for all values of k, i.e. $a_k \cdot b_k = a_1 \cdot b_1$ for all k. This check is equivalent to checking that $a_k^{-1} \cdot a_1 = b_k \cdot b_1^{-1}$ for all k and the latter check is done (in lines 12-16) by a pairwise comparison between every pair of parties, one from A (who holds $a_k^{-1} \cdot a_1$) and one from B (who holds $b_k b_1^{-1}$). If the tests pass then a_1, b_1 is taken to be the protocol output sharing, which is known to be correct, since one of the a_k, b_k have been correctly shared by the honest party in $S \cap T$, and $a_k \cdot b_k = a_1 \cdot b_1$ (and privacy is preserved since the parties only receive values they already have). Otherwise, if the test fails for some k, the players in A (resp. B) broadcast the values of $a_k^{-1} \cdot a_1$ (resp. $b_k b_1^{-1}$) and the values broadcast by the honest parties in A (resp. B) are compared. The values Hon_A^k (resp. Hon_B^k) broadcast by the honest parties can be deduced uniquely by the assumption that A (resp.B) cannot be covered by a union of two subsets in Δ (which in turn follows from the Δ-active-reliable property of coloring C). If they are equal, the test failure complaint was made falsely by a corrupted party, so it is ignored. Otherwise, if they are not equal, one of the parties P_k in $S \cap T$ must be corrupted. In this case, the corrupted parties already know both s and t so there is no privacy requirement, and the protocol *backtracks*: the parties in S (resp. T) broadcast the value of s (resp. t) to all parties, and output shares are defined to be $s \cdot t$ and 1, respectively, using the values of s and t broadcast by honest parties in S (resp. T).

Protocol 3 Shared 2-Product Protocol $\prod_S(\mathcal{G}_{grid}(\ell), C)$

Input: For $j = 1, \ldots, \ell$, parties P_i with $i \in \mathcal{O}_x(j)$ hold jth share $s_x(j) \in G$ of x and jth share $s_y(j) \in G$ of y, where $\mathbf{s}_x = (s_x(1), s_x(2), \ldots, s_x(\ell))$ and $\mathbf{s}_y = (s_y(1), s_y(2), \ldots, s_y(\ell))$ denote ℓ-of-ℓ sharing of $x \overset{\text{def}}{=} s_x(1) \cdot s_x(2) \cdots s_x(\ell)$ and $y \overset{\text{def}}{=} s_y(1) \cdot s_y(2) \cdots s_y(\ell)$, respectively. (We assume that C is a symmetric Δ-active-reliable n-coloring of $\mathcal{G}_{grid}(\ell)$, and define $\mathcal{O}_x(j) \overset{\text{def}}{=} C(1, j, V) = C(\ell+1-j, 1, H) = C(\ell+1, j, V)$).

```
1: Define input edge labels $v(\ell+1-j, 1, H) = s_x(j)$ and $v(1, j, V) = s_y(j)$ for $j \in [\ell]$.
2:
3: for $i = 1$ to $\ell$ do
4:
5:     for $j = 1$ to $\ell$ do
6:         Run protocol NodeMult with input share $s = v(i, j, H)$ held by party set
           $S = C(i, j, H)$ and input share $t = v(i, j, V)$ held by party set $T = C(i, j, V)$,
           and output party sets $A = C(i+1, j, V)$ and $B = C(i, j+1, H)$ if $j < \ell$ or
           $B = A$ if $j = \ell$. The protocol ends with output share $a$ held by party set $A$
           and output share $b$ held by party set $B$, with $a \cdot b = s \cdot t$.
7:         Define labels $v(i+1, j, V) \overset{\text{def}}{=} a$ for edge $(i+1, j, V)$ (or $v(i+1, j, V) = a \cdot b$ if
           $j = \ell$) and, if $j < \ell$, label $v(i, j+1, H) \overset{\text{def}}{=} b$ for edge $(i, j+1, H)$.
8:     end for
9: end for
```

Output: For $j = 1, \ldots, \ell$, parties P_i with $i \in \mathcal{O}_x(j)$ hold jth share $s_z(j) \overset{\text{def}}{=} v(\ell + 1, j, V) \in G$ of $z = x \cdot y$.

Lemma 2. *Assume that $S \cap T \notin \Delta$ and none of S, T, A, B are equal to the union of two sets from Δ. Then protocol **NodeMult**(s, t, S, T, A, B) satisfies* *the following properties, for all protocol inputs s, t, all $I \in \Delta$ and every active adversary $\mathcal{A}$ corrupting parties P_i for $i \in I$:*

- Correctness: *If, at the beginning of the protocol, all honest parties P_i with $i \in S$ (resp. $i \in T$) hold the same share s (resp. t), then at the end of the protocol, all honest parties P_i with $i \in A$ (resp. $i \in B$) hold the same share a (resp. b), with $a \cdot b = s \cdot t$.*
- Security: *Let $in^I_{\mathcal{A}} \subseteq \{s, t\}$ denote the protocol inputs given to $\mathcal{A}$, i.e. $s \in in^I_{\mathcal{A}}$ (resp. $t \in in^I_{\mathcal{A}}$) if $S \cap I \neq \emptyset$ (resp. if $T \cap I \neq \emptyset$). Similarly, let $out^I_{\mathcal{A}} \subseteq \{a, b\}$ denote the protocol outputs given to $\mathcal{A}$, i.e. $a \in out^I_{\mathcal{A}}$ (resp. $b \in out^I_{\mathcal{A}}$) if $A \cap I \neq \emptyset$ (resp. if $B \cap I \neq \emptyset$). Let $\mathcal{A}^{(s,t)}(in^I_{\mathcal{A}}, z_{aux})$ denote the output state of $\mathcal{A}$ on input $(in^I_{\mathcal{A}}, z_{aux})$ at the end of a run of **NodeMult**(s, t, S, T, A, B) (here z_{aux} is an auxiliary input representing the adversary's input state). Then, if $|out^I_{\mathcal{A}}| \leq 1$, there exists a probabilistic simulator algorithm S such that the random variables $\mathsf{REAL} \overset{\text{def}}{=} \langle \mathcal{A}^{(s,t)}(in^I_{\mathcal{A}}, z_{aux}), out^I_{\mathcal{A}} \rangle$ (representing the output state of $\mathcal{A}$ and protocol output given to $\mathcal{A}$) and $\mathsf{SIM} \overset{\text{def}}{=} \mathsf{S}(in^I_{\mathcal{A}}, z_{aux})$ (representing the simulated output state of $\mathcal{A}$ and protocol output given to $\mathcal{A}$) are identically distributed.*

3.3 Construction of a Δ-active-reliable coloring of graph $\mathcal{G}_{grid}(\ell)$

To complete our protocol construction, it remains to describe a Δ-active-reliable coloring of the graph $\mathcal{G}_{grid}(\ell)$. Our deterministic construction of such a coloring is given in Algorithm 5. It may be viewed as an adaptation of the deterministic t-reliable coloring of $\mathcal{G}_{grid}(\ell)$ from [9].

Lemma 3. *If Δ is Q^3, the coloring C returned by **DetCol** is a Δ-active-reliable n-coloring for $\mathcal{G}_{grid}(\ell)$.*

Proof. First, notice that $C(i, j, H) \cap C(i, j, V) = [n] \setminus (I_{\ell+1-i} \cup I_j)$ cannot be in Δ for any i, j since otherwise, it would imply that $[n]$ is the union of three sets $I_{\ell+1-i}, I_j, [n] \setminus (I_{\ell+1-i} \cup I_j)$ from Δ, contradicting the Q^3 property. Similarly, we must have $C(i, j, d)$ cannot be a union of two sets from Δ, otherwise again it would contradict the Q^3 property. For each $i \in [\ell]$, observe that the edges along the $(\ell+1-i)$th row and ith column of $\mathcal{G}_{grid}(\ell)$ are I_i-avoiding under the coloring C. The path $PATH_y$ for I_i is formed by the i'th column, while the path $PATH_x$ is formed by the portion of the $(\ell + 1 - i)$th row to the left of its intersection with $PATH_y$, with the rest of $PATH_x$ being the part of $PATH_y$ below the ith row. Finally, notice that $C(1, j, V) = C(\ell + 1 - j, 1, H) = C(\ell + 1, j, V)$ for all $j \in [\ell]$ so C is a symmetric Δ-active-reliable n-coloring, as claimed. □

Putting together the results of Lemma 1, Theorem 2 and Lemma 3, we get our main result.

Corollary 1. *If Δ is Q^3, there exists a Shared 2-Product black box Protocol for G satisfying active-correctness and strong Δ-security and with communication complexity $O(M(\Delta)^2 poly(n))$ group elements, a black box Δ-secure protocol for any G-circuit C with communication complexity $O(|C| \cdot M(\Delta)^2 \cdot poly(n))$ group elements.*

Protocol 4 **NodeMult**(s, t, S, T, A, B)

```
Input: Parties $P_i$ with $i \in S$ hold share $s \in G$, Parties $P_i$ with $i \in T$ hold share $t \in G$.
 1: Let $c = |S \cap T|$. Without loss of generality, assume $S \cap T = \{1, \dots, c\}$.
 2: for $k = 1$ to $c$ do
 3:   Party $P_k$ computes $u = s \cdot t$ (since $k \in S \cap T$, $P_k$ holds both $s$ and $t$).
 4:   Party $P_k$ chooses uniformly random $a_k, b_k \in G$ such that $a_k \cdot b_k = u$.
 5:   Party $P_k$ sends $a_k$ to each party $P_i$ with $i \in A$.
 6:   Every pair of parties $P_i$,$P_j$ with $i, j \in A$ send to each other the values $a_{k,i}$,$a_{k,j}$ of
      $a_k$ that $P_i$ (resp. $P_j$) received from $P_k$. If either $P_i$ or $P_j$ detects an inconsistency
      (i.e. $a_{k,i} \neq a_{k,j}$), it broadcasts a complaint against $P_k$.
 7:   If a complaint was broadcast in previous step against $P_k$, party $P_k$ broadcasts
      $a_k$ to all $n$ parties, and all parties accept this value as the correct value of $a_k$.
 8:   Party $P_k$ sends $b_k$ to each party $P_i$ with $i \in B$.
 9:   Every pair of parties $P_i$,$P_j$ with $i, j \in B$ send to each other the values $b_{k,i}$,$b_{k,j}$ of
      $b_k$ that $P_i$ (resp. $P_j$) received from $P_k$. If either $P_i$ or $P_j$ detects an inconsistency
      (i.e. $b_{k,i} \neq b_{k,j}$), it broadcasts a complaint against $P_k$.
10:   If a complaint was broadcast in previous step against $P_k$, party $P_k$ broadcasts
      $b_k$ to all $n$ parties, and all parties accept this value as the correct value of $b_k$.
11: end for
12: for $k = 2$ to $c$ do
13:   for all $i \in A$ and $j \in B$ do
14:     Party $P_i$ sends to $P_j$ the value $a_k^{-1} \cdot a_1$, and $P_j$ sends to $P_i$ the value $b_k \cdot b_1^{-1}$.
15:     If either $P_i$ or $P_j$ detects an inconsistency (i.e. $a_k^{-1} \cdot a_1 \neq b_k \cdot b_1^{-1}$), it broadcasts
        a complaint $k$.
16:   end for
17:   if a complaint $k$ was broadcast in previous step then
18:     All parties $P_i$ with $i \in A$ broadcast $a_k^{-1} \cdot a_1$ to all $n$ parties. Let $Hon_A^k$ denote
        the value $v$ such that all parties $P_i$ with $i \in A \setminus J$ broadcasted the value $v$, for
        some $J \in \Delta$. (such $v$ exists and is unique, see proof of Lemma 2).
19:     All parties $P_i$ with $i \in B$ broadcast $b_k \cdot b_1^{-1}$ to all $n$ parties. Let $Hon_B^k$ denote
        the value $v$ such that all parties $P_i$ with $i \in B \setminus J$ broadcasted the value $v$,
        for some $J \in \Delta$. (such $v$ exists and is unique, see proof of Lemma 2).
20:     if $Hon_A^k \neq Hon_B^k$ then
21:       Each party $P_i$ with $i \in S$ broadcasts $s$ to all $n$ parties. Let $Hon_s$ denote
          the value $v$ such that all parties $P_i$ with $i \in S \setminus J$ broadcasted the value $v$,
          for some $J \in \Delta$. (such $v$ exists and is unique, see proof of Lemma 2).
22:       Each party $P_i$ with $i \in T$ broadcasts $t$ to all $n$ parties. Let $Hon_t$ denote
          the value $v$ such that all parties $P_i$ with $i \in S \setminus J$ broadcasted the value $v$,
          for some $J \in \Delta$. (such $v$ exists and is unique, see proof of Lemma 2).
23:       return with each party $P_i$ with $i \in A$ holding output share $a = Hon_s \cdot$
          $Hon_t$, and each party $P_i$ with $i \in B$ holding output share $b = 1$.
24:     end if
25:   end if
26: end for
27: return with each party $P_i$ with $i \in A$ holding output share $a = a_1$, and each
    party $P_i$ with $i \in B$ holding output share $b = b_1$.
Output: Parties $P_i$ with $i \in A$ hold share $a \in G$, Parties $P_i$ with $i \in B$ hold share
    $b \in G$, with $a \cdot b = s \cdot t$.
```

```
Algorithm 5 Algorithm DetCol
Input: Graph $\mathcal{G}_{grid}(\ell)$ (see Fig. 1), where $\ell = M(\Delta)$.
  Let $I_1, \ldots, I_\ell$ denote the sequence of all maximal sets $\Delta$ (in some ordering).
  For $(i, j) \in [\ell+1] \times [\ell]$, $C(i, j, V) \overset{\text{def}}{=} [n] \setminus I_j$ and, if $i \leq \ell$, $C(i, j, H) \overset{\text{def}}{=} [n] \setminus I_{\ell+1-i}$.
Output: A $\Delta$-active-reliable coloring $C$ of Graph $\mathcal{G}_{grid}(\ell)$.
```

3.4 Application to Active-Secure General Multiparty Computation

In this Section, we explain how to apply our protocol for black-box groups to obtain a new approach for constructing actively-secure multiparty computation protocols for arbitrary Boolean circuits.

We begin by recalling a result of Barrington [1] that was used in the passive attack setting of [8] to reduce multiparty computation of arbitrary Boolean circuits to an S_5-circuit. Let C denote a G-circuit and let $f_C : G^m \to G$ be the function computed by C. Let 1_G denote the identity element of G. For some fixed $\sigma \in G \setminus \{1_G\}$, let $\phi_\sigma : \{0, 1\} \to G$ denote the encoding function mapping 0 to 1_G and 1 to σ. We say that a G-circuit C *computes* a Boolean function g if there exists $\sigma \in G$ such that $g(x_1, \ldots, x_n) = \phi_\sigma^{-1}(f_C(\phi_\sigma(x_1), \ldots, \phi_\sigma(x_n)))$ for all $(x_1, \ldots, x_n) \in \{0, 1\}^n$. Barrington's result can be stated as follows (see [8]).

Theorem 3 (Adapted from [1]). *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 $d' \leq 4d$.*

The S_5-circuit C' constructed in the proof of Theorem 3 computes the Boolean circuit C using the encoding function $\phi_\sigma : \{0, 1\} \to S_5$ mapping 0 to 1_{S_5} and 1 to the 5-cycle $\sigma = (1, 2, 3, 4, 5)$. In the passive attack setting of [8], all parties are assumed to honestly follow the protocol and correctly encode their Boolean inputs into the set $\{1_{S_5}, \sigma\}$, which are then used as input to the protocol for computing the S_5-circuit C'. However, in the active attack setting we study in this paper, one cannot directly apply our G-circuit protocol from the previous section to C', since the corrupted parties may choose as their inputs to circuit C' elements outside the set $\{1_{S_5}, \sigma\}$ in order to corrupt the protocol output. To fix this problem, we modify our protocol from the previous section for this application, by adding an additional input verification step. This verification step allows the parties to interactively check that the VSS'ed input elements of all parties are in the set $\{1_{S_5}, \sigma\}$, without revealing anything else about the shared inputs when they are indeed in the set $\{1_{S_5}, \sigma\}$. If a shared input is found by the check to be outside the set $\{1_{S_5}, \sigma\}$, the party who shared the input is declared corrupted, and the corresponding shared input is redefined to be a VSS sharing of the default value 1_{S_5}. The correctness of the test in Protocol 6 is shown by Lemma 4. It uses elementary properties of the group S_5, and its proof can be found in the full version of the paper.

Protocol 6 Verification Step (inserted into Protocol $\prod_a(C, \prod_S)$.

```
Input: For $i \in [m]$, the parties hold a VSS sharing $\mathbf{s}_{x_i} = (s_{x_i}(1), \ldots, s_{x_i}(\ell))$ of input
    $x_i \in S_5$ shared by party $P_{j(i)}$, where, for each $j \in [\ell]$, share $s_{x_i}(j)$ is held by players
    in set $\mathcal{O}_x(j) = [n] \setminus I_j$.
    for $i = 1$ to $m$ do
        1. The parties jointly compute, using $G$-circuit protocol from Sec. 3.1 on the
        VSS'ed input $x_i \in S_5$, the value $y_1 = E_1(x_i)$, where $E_1(x) = x \cdot \sigma \cdot x^{-1} \cdot \sigma^{-1}$.
        2. The parties jointly compute, using $G$-circuit protocol from Sec. 3.1 on the
        VSS'ed input $x_i \in S_5$, the value $y_2 = E_2(x_i)$, where $E_2(x) = x \cdot g_1 \cdot x^2 \cdot g_2 \cdot x^3 \cdot (g_1 \cdot g_2)^{-1}$, $g_1 = (1)(2,3)(4)(5)$ and $g_2 = (1,2,5,4,3)$.
        3. If $y_1 = 1_{S_5}$ and $y_2 = 1_{S_5}$, the parties conclude that $x_i \in \{1_{S_5}, \sigma\}$. Else, the
        parties conclude that $x_i \notin \{1_{S_5}, \sigma\}$, declare party $P_{j(i)}$ as corrupted, and set
        $x_i = 1_{S_5}$, with all VSS shares $s_{x_i}(j) = 1_{S_5}$ for $j \in [\ell]$.
    end for
Output: For $i \in [m]$, the parties hold a VSS sharing $\mathbf{s}_{x_i} = (s_{x_i}(1), \ldots, s_{x_i}(\ell))$ of input
    $x_i \in S_5$ with $x_i \in \{1_{S_5}, \sigma\}$ and for $j \in [\ell]$, share $s_{x_i}(j)$ is held by set $\mathcal{O}_x(j) = [n] \setminus I_j$.
```

Lemma 4. *For each $i \in [m]$, the tests $y_1 = 1_{S_5}$ and $y_2 = 1_{S_5}$ are both verified if and only if $x_i \in \{1_{S_5}, \sigma\}$.*

By adding the verification Protocol 6 to our protocol in Sec. 3.1 and applying it to the S_5-circuit C' produced by Theorem 3, we obtain an active-secure protocol for computing any Boolean function. The correctness follows from Lemma 4 and the correctness of our G-circuit protocol, and the security follows from the simulatability of our G-circuit protocol. The proof follows by a straightforward modification of the proof of Lemma 1 and is omitted.

Corollary 2. *If Δ is Q^3, the above protocol is a Δ-secure protocol for any Boolean circuit C, with communication complexity $O(|C| \cdot M(\Delta)^2 \cdot poly(n))$ bits.*

Our protocol works quite differently from previous approaches to general secure multiparty computation that work over a field. The latter can achieve a similar communication complexity of $O(M(\Delta)^2)$ bits [20]. Because our protocol only runs the full VSS sharing protocol at the beginning but not at each intermediate graph node, its communication complexity is only $O(\ell^2 \cdot \text{poly}(n))$ group elements for multiplying two VSSed group elements whereas the complexity of the field-based protocol of Maurer [19], which is also based on Maurer's VSS, is $O(\ell^3 \cdot \text{poly}(n))$ field elements for multiplying two VSSed field elements where $\ell = M(\Delta)$, i.e. our protocol saves a factor of order $\Omega(\ell)$ (ignoring the dependance on n) in communication complexity over Maurer's protocol when applied to computing the same Boolean circuit C (with Maurer's protocol over $GF(2)$ and our protocol over S_5).

Acknowledgements. We thank Huaxiong Wang for encouraging us to address the problem. We also thank Yuval Ishai for pointing out the problem (addressed in Sec. 3.4) of incorrectly encoded inputs in applying our protocol to the computation of Boolean circuits. Part of this work was done while Y. Desmedt was visiting Macquarie University, funded by BT as "BT Chair of Information

Security". The research of J. Pieprzyk and R. Steinfeld was supported by an Australian Research Fellowship (ARF) from the Australian Research Council (ARC), and ARC Discovery Grants DP0987734 and DP110100628.

References

1. D.A. Barrington. Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1. In *STOC '86*, pages 1–5, 1986.
2. Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure mpc with linear communication complexity. In *TCC 2008*, pages 213–230, 2008.
3. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *STOC '88*, pages 1–10, 1988.
4. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *STOC '88*, pages 11–19, 1988.
5. R. Cramer, S. Fehr, Y. Ishai, and E. Kushilevitz. Efficient Multi-Party Computation Over Rings. In *EUROCRYPT 2003*, pages 596–613, 2003.
6. R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT 2000*, pages 316–334.
7. I. Damgård, Y. Ishai, and M. Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *EUROCRYPT 2010*, pages 445–465, 2010.
8. Y. Desmedt, J. Pieprzyk, R. Steinfeld, X. Sun, C. Tartary, H. Wang, and A. C.-C. Yao. Graph coloring applied to secure computation in non-abelian groups. *J. Cryptology*, 2011. To Appear.
9. Y. Desmedt, J. Pieprzyk, R. Steinfeld, and H. Wang. On Secure Multiparty Protocols in Black Box Groups. In *CRYPTO 2007*, pages 591–612, 2007.
10. M. Fitzi, M. Hirt, and U. Maurer. Trading correctness for privacy in unconditional multi-party computation. In *CRYPTO 1998*, pages 121–136, 1998.
11. M. Fitzi, M. Hirt, and U. Maurer. General adversaries in unconditional multi-party computation. In *ASIACRYPT 1999*, pages 232–246, 1999.
12. M. Fitzi and U. Maurer. Efficient byzantine agreement secure against general adversaries. In *DISC '98*, pages 134–148, 1998.
13. O. Goldreich. *Foundations of Cryptography: Volume II - Basic Applications.* Cambridge University Press, 2004.
14. O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game. In *STOC '87*, pages 218–229, 1987.
15. M. Hirt and U. Maurer. Complete Characterization of Adversaries Tolerable in Secure Multi-Party Computation. In *PODC '97*, pages 25–34, 1997.
16. M. Hirt and U. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptology*, 13(1):31–60, 2000.
17. M. Hirt and U. Maurer. Robustness for free in unconditional multi-party computation. In *CRYPTO 2001*, pages 101–118, 2001.
18. M. Hirt, U. Maurer, and B. Przydatek. Efficient secure multi-party computation. In *ASIACRYPT 2000*, pages 143–161, 2000.
19. U. Maurer. Secure multi-party computation made simple. *Discrete Applied Mathematics*, 154:370–381, 2006.
20. B. Prabhu, K. Srinathan, and C. Pandu Rangan. Trading players for efficiency in unconditional multiparty computation. In *SCN 2002*, pages 342–353, 2002.
21. X. Sun, A. C.-C. Yao, and C. Tartary. Graph design for secure multiparty computation over non-Abelian groups. In *ASIACRYPT 2008*, pages 37–53, 2008.