# Secure Multiparty Computation from Graph Colouring

Ron Steinfeld
Monash University

July 2012

# Acknowledgements

Based on joint work with (subsets of):
Yvo Desmedt, Josef Pieprzyk, Huaxiong Wang, Xiaoming Sun, Christophe Tartary, Andrew Chi-Chih Yao

# Outline

- **The Problem**: Secure multiparty computation in black-box groups
  - Motivation / definition
  - Attack model (computationally unbounded, passive)
  - Previous approaches
- **Our Results**:
  - Reduction: $n$-Product to Shared 2-Product
  - Reduction: Shared 2-Product to $t$-Reliable Planar Graph Colouring
  - Constructions of $t$-Reliable Planar Graph Colourings
  - Extensions (briefly):
    - Computing arbitrary functions
    - Security against active adversaries
- **Open Problems**

# What is secure multiparty computation?

Typical example: Electronic Auction

- $n$ parties: $P_1, \ldots, P_n$
- Each $P_i$ commits his bid $x_i \in \mathbb{N}$.
- At the end, the highest bidder wins auction

**Basic requirements (informal)**:

- Correctness: All parties learn the winning bid / bidder :

$$f(x_1, \ldots, x_n) = (\max_i x_i, \arg\max_i x_i)$$

- Privacy: No party learns anything about losing bids, except what is leaked by winning bid.

# What is secure multiparty computation?

Typical example: Electronic Auction

- $n$ parties: $P_1,\ldots,P_n$
- Each $P_i$ commits his bid $x_i \in \mathbb{N}$.
- At the end, the highest bidder wins auction

**Basic requirements (informal)**:

- Correctness: All parties learn the winning bid / bidder :

$$f(x_1,\ldots,x_n) = (\max_i x_i, \arg\max_i x_i)$$

- Privacy: No party learns anything about losing bids, except what is leaked by winning bid.

# What is secure multiparty computation?

Typical example: Electronic Auction

- $n$ parties: $P_1, \ldots, P_n$
- Each $P_i$ commits his bid $x_i \in \mathbb{N}$.
- At the end, the highest bidder wins auction

**Basic requirements (informal)**:

- Correctness: All parties learn the winning bid / bidder :

$$f(x_1, \ldots, x_n) = (\max_i x_i, \arg\max_i x_i)$$

- Privacy: No party learns anything about losing bids, except what is leaked by winning bid.

# What is secure multiparty computation?

How to achieve this?

If we live in an ideal world: use a Trusted Party (TP)

- TP serves as the auctioneer
- Each $P_i$ sends his bid $x_i \in \mathbb{N}$ to TP
- TP privately computes and announces $(\max_i x_i, \arg\max_i x_i)$ to all $P_i$'s

What if, in real world, such a TP does not exist?

Possible answer: $t$-private secure multiparty computation

- Parties run a distributed computation protocol among themselves
  - Every pair of parties can communicate privately from all other parties
- At protocol end, all parties can compute result $f(x_1, \ldots, x_n)$.
- Privacy holds as long as not more than $t$ parties collude

# What is secure multiparty computation?

How to achieve this?

If we live in an ideal world: use a Trusted Party (TP)

- TP serves as the auctioneer
- Each $P_i$ sends his bid $x_i \in \mathbb{N}$ to TP
- TP privately computes and announces $(\max_i x_i, \arg\max_i x_i)$ to all $P_i$'s

What if, in real world, such a TP does not exist?

Possible answer: $t$-private secure multiparty computation

- Parties run a distributed computation protocol among themselves
  - Every pair of parties can communicate privately from all other parties
- At protocol end, all parties can compute result $f(x_1, \ldots, x_n)$.
- Privacy holds as long as not more than $t$ parties collude

# What is secure multiparty computation?

How to achieve this?

If we live in an ideal world: use a Trusted Party (TP)

- TP serves as the auctioneer
- Each $P_i$ sends his bid $x_i \in \mathbb{N}$ to TP
- TP privately computes and announces $(\max_i x_i, \arg\max_i x_i)$ to all $P_i$'s

What if, in real world, such a TP does not exist?

Possible answer: $t$-private secure multiparty computation

- Parties run a distributed computation protocol among themselves
  - Every pair of parties can communicate privately from all other parties
- At protocol end, all parties can compute result $f(x_1, \ldots, x_n)$.
- Privacy holds as long as not more than $t$ parties collude

# What is secure multiparty computation?

How to achieve this?

If we live in an ideal world: use a Trusted Party (TP)

- TP serves as the auctioneer
- Each $P_i$ sends his bid $x_i \in \mathbb{N}$ to TP
- TP privately computes and announces $(\max_i x_i, \arg\max_i x_i)$ to all $P_i$'s

What if, in real world, such a TP does not exist?

Possible answer: $t$-private secure multiparty computation

- Parties run a distributed computation protocol among themselves
  - Every pair of parties can communicate privately from all other parties
- At protocol end, all parties can compute result $f(x_1, \ldots, x_n)$.
- Privacy holds as long as not more than $t$ parties collude

# Secure Multiparty computation: attack model

Several possible flavours of security, depending on:

- Computational abilities
  - Computationally bounded: security only guaranteed if attack computing time $\leq$ (large) bound $T$.
  - Computationally unbounded ('information theoretic'): security holds regardless of attack computation time.
- Allowed deviation from prescribed protocol
  - Passive attacks ('Honest But Curious'): colluding parties follow protocol, but analyze protocol messages they receive to learn about other party's inputs.
  - Active attacks: colluding parties can misbehave arbitrarily, to disrupt correctness and/or breach privacy of other parties

Focus on computationally unbounded, passive attacks (at end: a little on active security).

# Secure Multiparty computation: attack model

Several possible flavours of security, depending on:

- Computational abilities
  - Computationally bounded: security only guaranteed if attack computing time $\leq$ (large) bound $T$.
  - Computationally unbounded ('information theoretic'): security holds regardless of attack computation time.
- Allowed deviation from prescribed protocol
  - Passive attacks ('Honest But Curious'): colluding parties follow protocol, but analyze protocol messages they receive to learn about other party's inputs.
  - Active attacks: colluding parties can misbehave arbitrarily, to disrupt correctness and/or breach privacy of other parties

Focus on computationally unbounded, passive attacks (at end: a little on active security).

# Secure Multiparty computation: attack model

Several possible flavours of security, depending on:

- Computational abilities
  - Computationally bounded: security only guaranteed if attack computing time $\leq$ (large) bound $T$.
  - Computationally unbounded ('information theoretic'): security holds regardless of attack computation time.
- Allowed deviation from prescribed protocol
  - Passive attacks ('Honest But Curious'): colluding parties follow protocol, but analyze protocol messages they receive to learn about other party's inputs.
  - Active attacks: colluding parties can misbehave arbitrarily, to disrupt correctness and/or breach privacy of other parties

Focus on computationally unbounded, passive attacks (at end: a little on active security).

# Secure Multiparty computation: attack model

Several possible flavours of security, depending on:

- Computational abilities
  - Computationally bounded: security only guaranteed if attack computing time $\leq$ (large) bound $T$.
  - Computationally unbounded ('information theoretic'): security holds regardless of attack computation time.
- Allowed deviation from prescribed protocol
  - Passive attacks ('Honest But Curious'): colluding parties follow protocol, but analyze protocol messages they receive to learn about other party's inputs.
  - Active attacks: colluding parties can misbehave arbitrarily, to disrupt correctness and/or breach privacy of other parties

Focus on computationally unbounded, passive attacks (at end: a little on active security).

# Our Problem: Secure Product in Black-Box Groups

Fix a finite group $G$. For $i = 1, \ldots, n$ party $P_i$ holds input $x_i \in G$. Our goal - a secure $n$-Party protocol for computing $n$-Product function over $G$:

$$f_G(x_1, \ldots, x_n) = x_1 \cdots x_n.$$

Our protocols treat $G$ as a black-box – the only computations allowed in the protocol are:

- Group operation: $(x, y) \in G^2 \mapsto x \cdot y \in G$
- Group inverse: $x \in G \mapsto x^{-1} \in G$
- Sampling a uniformly random element of $G$

At end: secure computation of any function by reduction to (a variant of) our problem over $G = S_5$.

## Our Problem: Secure Product in Black-Box Groups

Fix a finite group $G$. For $i = 1, \ldots, n$ party $P_i$ holds input $x_i \in G$. Our goal - a secure $n$-Party protocol for computing $n$-Product function over $G$:

$$f_G(x_1, \ldots, x_n) = x_1 \cdots x_n.$$

Our protocols treat $G$ as a black-box – the only computations allowed in the protocol are:

- Group operation: $(x, y) \in G^2 \mapsto x \cdot y \in G$
- Group inverse: $x \in G \mapsto x^{-1} \in G$
- Sampling a uniformly random element of $G$

At end: secure computation of any function by reduction to (a variant of) our problem over $G = S_5$.

# Our Problem: Secure Product in Black-Box Groups

Fix a finite group $G$. For $i = 1, \ldots, n$ party $P_i$ holds input $x_i \in G$. Our goal - a secure $n$-Party protocol for computing $n$-Product function over $G$:

$$f_G(x_1, \ldots, x_n) = x_1 \cdots x_n.$$

Our protocols treat $G$ as a black-box – the only computations allowed in the protocol are:

- Group operation: $(x, y) \in G^2 \mapsto x \cdot y \in G$
- Group inverse: $x \in G \mapsto x^{-1} \in G$
- Sampling a uniformly random element of $G$

At end: secure computation of any function by reduction to (a variant of) our problem over $G = S_5$.

# Secure Multiparty computation: attack model

Precise formulation of $t$-privacy of protocol $\prod$:
Let

- Inputs be $\vec{x} = (x_1, \ldots, x_n)$,
- $\mathrm{VIEW}_I^{\prod}(\vec{x})$ denote protocol view of parties in subset $I \subseteq [n]$.
- $\vec{x}_I$ denotes the inputs of parties in $I$.
- Protocol output $y = f_G(x_1, \ldots, x_n) = x_1 \cdots x_n$.

## Definition

$\prod$ is a $t$-private protocol for computing $f_G$ if there exists a probabilistic polynomial-time algorithm S, such that, for every $I \subset [n]$ with $|I| \leq t$ and every $(x_1, \ldots, x_n) \in G^n$, the random variables $\mathrm{S}(I, \vec{x}_I, y)$ and $\mathrm{VIEW}_I^{\prod}(\vec{x})$ are identically distributed.

## Some background and related work

- Research on secure computation began in early 1980's: Yao's Millionaire problem
- General result by end of '80's
  - **Theorem (Cramer et al '88, Ben-Or et al '88):** Any function $f : (\{0,1\}^{\ell_i})^n \rightarrow \{0,1\}^{\ell_o}$ can be $t$-privately computed by an $n$-party protocol (in the passive, computationally unbounded model) if and only if $t < n/2$. The protocol communication complexity is $O(\mathcal{P}oly(n) \cdot |C|)$, where $C$ is a boolean circuit computing $f$.
- These protocols reduce to a computation over a finite field:
  - $f$ is expressed as a Boolean circuit $C$ (i.e. an arithmetic circuit over finite field $\mathbb{F}_2$).
  - Secret inputs shared over a finite field $\mathbb{F}_q$ among $n$ parties (using Shamir's $(t+1)$-of-$n$ threshold secret sharing scheme).
  - At each AND gate of $C$, use Shamir multiplicative property to multiply shared inputs to shared output (resharing also needed)

## Some background and related work

- Research on secure computation began in early 1980's: Yao's Millionaire problem
- General result by end of '80's
  - **Theorem (Cramer et al '88, Ben-Or et al '88):** Any function $f : (\{0,1\}^{\ell_i})^n \rightarrow \{0,1\}^{\ell_o}$ can be $t$-privately computed by an $n$-party protocol (in the passive, computationally unbounded model) if and only if $t < n/2$. The protocol communication complexity is $O(\mathcal{P}oly(n) \cdot |C|)$, where $C$ is a boolean circuit computing $f$.
- These protocols reduce to a computation over a finite field:
  - $f$ is expressed as a Boolean circuit $C$ (i.e. an arithmetic circuit over finite field $\mathbb{F}_2$).
  - Secret inputs shared over a finite field $\mathbb{F}_q$ among $n$ parties (using Shamir's $(t+1)$-of-$n$ threshold secret sharing scheme).
  - At each AND gate of $C$, use Shamir multiplicative property to multiply shared inputs to shared output (resharing also needed)

# Some background and related work

- Research on secure computation began in early 1980's: Yao's Millionaire problem
- General result by end of '80's
  - **Theorem (Cramer et al '88, Ben-Or et al '88):** Any function $f : (\{0,1\}^{\ell_i})^n \rightarrow \{0,1\}^{\ell_o}$ can be $t$-privately computed by an $n$-party protocol (in the passive, computationally unbounded model) if and only if $t < n/2$. The protocol communication complexity is $O(\mathcal{P}oly(n) \cdot |C|)$, where $C$ is a boolean circuit computing $f$.
- These protocols reduce to a computation over a finite field:
  - $f$ is expressed as a Boolean circuit $C$ (i.e. an arithmetic circuit over finite field $\mathbb{F}_2$).
  - Secret inputs shared over a finite field $\mathbb{F}_q$ among $n$ parties (using Shamir's $(t+1)$-of-$n$ threshold secret sharing scheme).
  - At each AND gate of $C$, use Shamir multiplicative property to multiply shared inputs to shared output (resharing also needed)

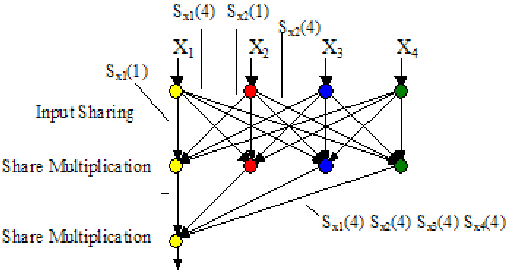
# Secure Product in a Group: Abelian case (folklore)

- Efficient Black-Box Protocol for **Abelian** Groups ($t < n$)
  - Building Block: $n$-of-$n$ secret sharing over **Abelian** group $G$:

$$x = s_x(1) \cdot s_x(2) \cdots s_x(n).$$

  - **Abelian $G$ implies Multiplicative Property:**

$$x \cdot y = s_x(1) \cdots s_x(n) \cdot s_y(1) \cdots s_y(n) = s_x(1) \cdot s_y(1) \cdots s_x(n) \cdot s_y(n)$$



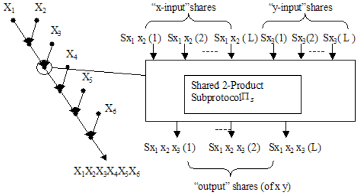
# Secure Computation of $n$-Product

How to extend the Abelian protocol to Non-Abelian groups?

Order is important: for correctness in non-Abelian $G$, restrict to **planar** communication graphs

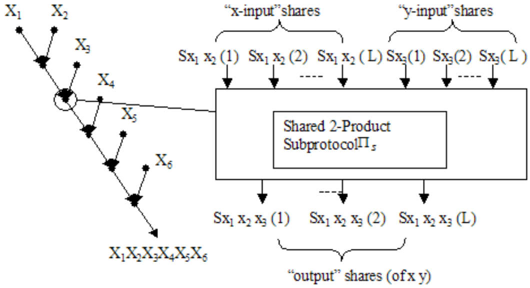
# Constructions, Step 1: $n$-Product to 2-Product

- Reducing $n$-Product to Shared 2-Product:
  - Use binary tree for computing $y = x_1 \cdots x_n$ from $x_1, \ldots, x_n$, with $x_i$'s at leaves, and product at each internal node.
  - Input Sharing: $i$th party shares $x_i$ to $\ell$ parties according to sharing functions $\mathcal{O}_x$, $\mathcal{O}_y$ of subprotocol $\Pi_S$.
  - For each internal node of tree, invoke instance of subprotocol $\Pi_S$ to multiply shared inputs to a shared outputs.
  - Obtain shared root value $y = x_1 \cdots, x_n$. Shares $s_z(1), \ldots, s_z(\ell)$ broadcast to all parties, who compute $y = s_z(1) \cdots s_z(\ell)$.



# Constructions, Step 1: $n$-Product to Shared 2-Product

- To get $t$-privacy of $n$-Product protocol $\Pi$, require **strong $t$-privacy** for Shared 2-product subprotocol $\Pi_S$:
  - For each $t$-collusion $I$, given:
    - All '$x$-input' shares except one not held by $I$ ($j^*$th share)
    - All '$y$-input' shares except one not held by $I$ ($j_y^*$th share)
  - It is possible to simulate internal view and all output shares except one not held by $I$ ($j^*$th share).

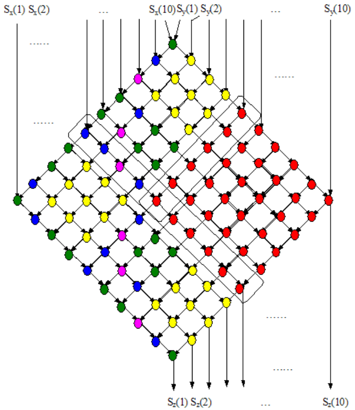


# Constructions, Step 1: $n$-Product to Shared 2-Product

- **Lemma.** For any binary computation tree for $f_G$, if Shared 2-product subprotocol $\Pi_S$ satisfies strong $t$-privacy, then $n$-Product protocol $\Pi$ is $t$-private.
- Proof Idea:
  - For each collusion $I$ (by $\ell$-of-$\ell$ property of input sharing), all $\ell - 1$ except one share of each $x_i$ can be simulated by a $t$-collusion $I$.
  - At each internal node of the tree, apply simulator for $\Pi_S$ to simulate view of $I$ in corresponding subprotocol run and $\ell - 1$ output shares (use as $x$-input shares to following simulator run).
  - Finally get simulated $\ell - 1$ shares of output value (root node), and compute remaining $\ell$th share from known $\ell - 1$ shares and the given protocol output $y$.

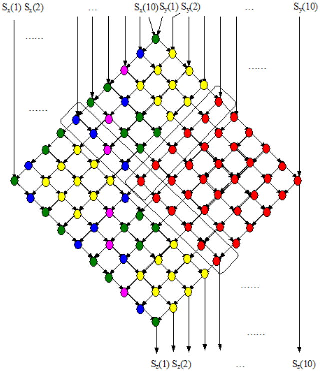
# Constructions, Step 2: 2-Product from Graph Colouring

- Use planar communication graphs which preserve product at each row - **Admissible PDAGs** (Planar Directed Acyclic Graphs).



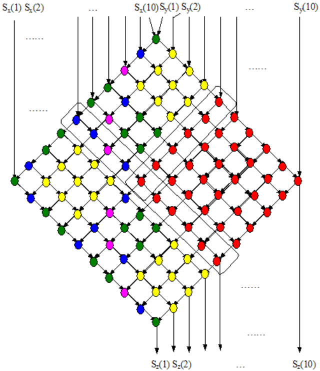
# Constructions, Step 2: 2-Product from Graph Colouring

- Q. Which $n$-Colourings of a given graph give strong $t$-privacy?
- A. $t$-**reliable** $n$-**colouring:** For each $t$-collusion $I$, there is:
  - An $I$-avoiding path from $j^*$th $x$-input to $j^*$th output
  - An $I$-avoiding path from $j_y$th $y$-input to $j^*$th output



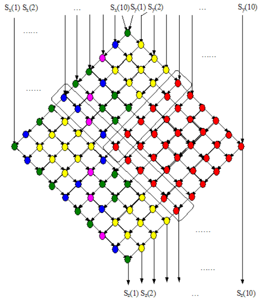
# Constructions, Step 2: 2-Product from Graph Colouring

- Q. Which $n$-Colourings of a given graph give strong $t$-privacy?
- A. $t$-**reliable** $n$-**colouring:** For each $t$-collusion $I$, there is:
  - An $I$-avoiding path from $j^*$th $x$-input to $j^*$th output
  - An $I$-avoiding path from $j_y$th $y$-input to $j^*$th output



# Constructions, Step 2: 2-Product from Graph Colouring

- **Lemma.** If $\mathcal{G}$ is an admissible PDAG and $C$ is a $t$-Reliable $n$-Colouring for $\mathcal{G}$ then $\Pi_S(\mathcal{G}, C)$ achieves strong $t$-privacy.
- Proof Idea: At each node along path, one outgoing share is not in collusion's view; remaining $k-1$ shares are random and independent of the node value (proof extends also to paths with upward edges).



# Step 3: Realizing $t$-reliable $n$-Colourings

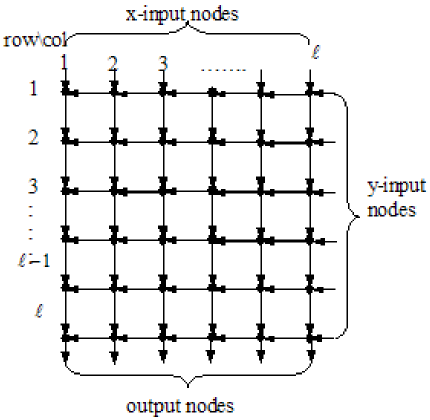
Two constructions:

- Deterministic: $t < n/2$ optimal, but size $\ell$ exponential in $n$
- Probabilistic: $t < n/2$ optimal, size $\ell = \mathcal{P}oly(n)$, but error probability $\delta$ exponentially small in $n$.

Recursive deterministic construction [Sun et al, 2008] trades off resilience $t < n^{1-\varepsilon}$ for smaller size $\ell = O(\mathcal{P}oly(n))$.

## Step 3: $t$-reliable $n$-Colourings – Deterministic construction

- We consider the $\ell \times \ell$ square admissible PDAG $\mathcal{G}_{tri}(\ell, \ell)$.



# Step 3: $t$-Reliable $n$-Colourings – Deterministic Construction

- Example colouring, $n = 5$, $t = 2$, $\ell = \binom{5}{2} = 10$.

| | {1,2} | {1,3} | {1,4} | {1,5} | {2,3} | {2,4} | {2,5} | {3,4} | {3,5} | {4,5} |
|---|---|---|---|---|---|---|---|---|---|---|
| {1,2} | 3 | 4 | 3 | 3 | 4 | 3 | 3 | 5 | 4 | 3 |
| {1,3} | 4 | 2 | 2 | 2 | 4 | 5 | 4 | 2 | 2 | 2 |
| {1,4} | 3 | 2 | 2 | 2 | 5 | 3 | 3 | 2 | 2 | 2 |
| {1,5} | 3 | 2 | 2 | 2 | 4 | 3 | 3 | 2 | 2 | 2 |
| {2,3} | 4 | 4 | 5 | 4 | 1 | 1 | 1 | 1 | 1 | 1 |
| {2,4} | 3 | 5 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 1 |
| {2,5} | 3 | 4 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 1 |
| {3,4} | 5 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| {3,5} | 4 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| {4,5} | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |

# Step 3: $t$-Reliable $n$-Colourings – Deterministic Construction

Generalisation to any $n, t$ gives:

**Lemma**

*For $t < n/2$, $C_{comb}$ is a Symmetric $t$-Reliable $n$-Colouring for graph $\mathcal{G}_{tri}(\ell, \ell)$, with $\ell = \binom{n}{t}$.*

**Corollary**

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

**Remark.** The condition $t < n/2$ is necessary for existence of a $t$-reliable $n$-coloring:

- If $n = 2t$, an $I$-avoiding top-bottom path contains $\leq |[n] \setminus I| = 2t - t = t$ colours – it is a left-right cutset!

# Step 3: $t$-Reliable $n$-Colourings – Deterministic Construction

Generalisation to any $n, t$ gives:

**Lemma**

*For $t < n/2$, $C_{comb}$ is a Symmetric $t$-Reliable $n$-Colouring for graph $\mathcal{G}_{tri}(\ell, \ell)$, with $\ell = \binom{n}{t}$.*

**Corollary**

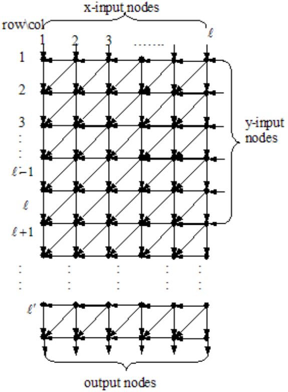
*For any $t < n/2$, there exists a black-box $t$-private protocol for $f_G$ with communication complexity $O(n\binom{2t+1}{t}^2)$ group elements.*

**Remark.** The condition $t < n/2$ is <u>necessary</u> for existence of a $t$-reliable $n$-coloring:

- If $n = 2t$, an $I$-avoiding top-bottom path contains $\leq |[n] \setminus I| = 2t - t = t$ colours – it is a left-right cutset!

# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

- We add diagonal edges and allow for rectangular $\ell' \times \ell$ admissible PDAG $\mathcal{G}_{tri}(\ell', \ell)$.



# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

- Question: Can we get $t$-Reliable $n$-Colourings with $\ell$ polynomial in $t$?
- YES - use a random colouring!
- Actually, we will show a random colouring is only **weakly** $t$-Reliable, i.e. for each $t$-colour subset $I \subset [n]$:
  - There exists an $I$-avoiding top-bottom path $P_x$
  - There exists an $I$-avoiding right-left path $P_y$
  - No need to worry about matching entry and exit positions

# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

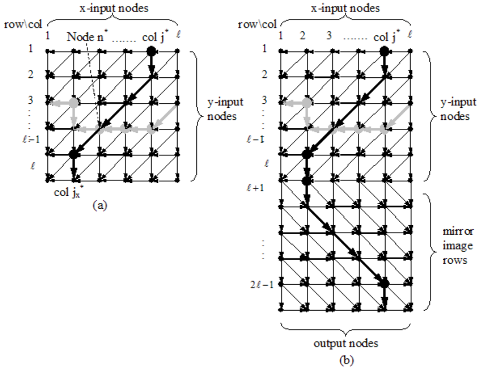
- Question: Can we get $t$-Reliable $n$-Colourings with $\ell$ polynomial in $t$?
- YES - use a random colouring!
- Actually, we will show a random colouring is only **weakly** $t$-Reliable, i.e. for each $t$-colour subset $I \subset [n]$:
  - There exists an $I$-avoiding top-bottom path $P_x$
  - There exists an $I$-avoiding right-left path $P_y$
  - No need to worry about matching entry and exit positions

# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

- Question: Can we get $t$-Reliable $n$-Colourings with $\ell$ polynomial in $t$?
- YES - use a random colouring!
- Actually, we will show a random colouring is only **weakly** $t$-Reliable, i.e. for each $t$-colour subset $I \subset [n]$:
  - There exists an $I$-avoiding top-bottom path $P_x$
  - There exists an $I$-avoiding right-left path $P_y$
  - No need to worry about matching entry and exit positions

# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

- **Lemma (Mirror).** Any weakly $t$-Reliable $n$-Colouring for PDAG $\mathcal{G}_{tri}(\ell, \ell)$ can be converted into a (standard) $t$-Reliable $n$-Colouring for a rectangular admissible PDAG $\mathcal{G}_{gtri}(2\ell - 1, \ell)$.



# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

**Goal**: Find an upper bound on error probability $\delta$ that $C_{rand}$ is not weakly $t$-Reliable.

Link with percolation theory!

Fix collusion $I \subset [n]$ with $|I| = t$. Since we use a uniformly random $n$-colouring:

- Each node of graph is in $I$ ('closed') with probability $p = t/n$.
- Want to upper bound probability that there is no open top-bottom path in graph

**Observation ("Self-Duality" Property of $T$):** For triangular lattice $\mathcal{G}_{tri}(\ell, \ell)$, there is no open top-bottom path iff there is a closed left-right 'cutting' path.

So, suffices to upper bound the probability of a closed left-right path.

# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

**Goal**: Find an upper bound on error probability $\delta$ that $C_{rand}$ is not weakly $t$-Reliable.

Link with percolation theory!

Fix collusion $I \subset [n]$ with $|I| = t$. Since we use a uniformly random $n$-colouring:

- Each node of graph is in $I$ ('closed') with probability $p = t/n$.
- Want to upper bound probability that there is no open top-bottom path in graph

**Observation ("Self-Duality" Property of $T$):** For triangular lattice $\mathcal{G}_{tri}(\ell, \ell)$, there is no open top-bottom path iff there is a closed left-right 'cutting' path.

So, suffices to upper bound the probability of a closed left-right path.

# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

**Goal**: Find an upper bound on error probability $\delta$ that $C_{rand}$ is not weakly $t$-Reliable.

Link with percolation theory!

Fix collusion $I \subset [n]$ with $|I| = t$. Since we use a uniformly random $n$-colouring:

- Each node of graph is in $I$ ('closed') with probability $p = t/n$.
- Want to upper bound probability that there is no open top-bottom path in graph

**Observation ("Self-Duality" Property of $T$):** For triangular lattice $\mathcal{G}_{tri}(\ell, \ell)$, there is no open top-bottom path iff there is a closed left-right 'cutting' path.

So, suffices to upper bound the probability of a closed left-right path.

# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

Percolation theory result, for the infinite triangular lattice $T$.

**Theorem (Hammersely '57)**

*Fix node $n$ of $T$. If each node closed independently with prob. $p$, there exists a critical prob. $p_c(T)$ such that, for for $p < p_c(T)$,*

$$\Pr[\exists \textit{ a closed path in } T \textit{ of length } \ell \textit{ starting at } n] < \exp(-\ell/r(p)),$$

*where $r(p)$ depends on $p$ but not on $\ell$. Moreover, $p_c(T) = 1/2$.*

In our case, $p = t/n$. If $t/n = \frac{1}{2+\varepsilon}$ for some constant $\varepsilon > 0$,

$$\delta = \Pr(C_{rand} \text{ is bad}) \leq 2 \cdot \binom{n}{t} \cdot \ell \cdot \exp(-(\ell - 1)/r(\varepsilon)),$$

so can use $\ell = O(n + \log \delta^{-1})$, for any desired error probability $\delta$.

# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

In the optimal case, $t/n = 1/2 - \frac{1}{2n} = p_c(T) - o(1)$, we are in the 'near-critical' percolation region.

The function $r(p)$ seems not so well understood for general graphs in this region...

But for the triangular lattice $T$, celebrated results of [Smirnov,Werner 2001] can be used to show

$$r(p) \to c \cdot (p - 1/2)^{-91/36+o(1)} \text{ as } p \to 1/2,$$

which implies that we can take $\ell = O(n^{91/36+\varepsilon} \cdot (n + \log(\delta^{-1}))$ for error probability $\delta$.

# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

In the optimal case, $t/n = 1/2 - \frac{1}{2n} = p_c(T) - o(1)$, we are in the 'near-critical' percolation region.

The function $r(p)$ seems not so well understood for general graphs in this region...

But for the triangular lattice $T$, celebrated results of [Smirnov,Werner 2001] can be used to show

$$r(p) \to c \cdot (p - 1/2)^{-91/36+o(1)} \text{ as } p \to 1/2,$$

which implies that we can take $\ell = O(n^{91/36+\varepsilon} \cdot (n + \log(\delta^{-1}))$ for error probability $\delta$.

# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

In the optimal case, $t/n = 1/2 - \frac{1}{2n} = p_c(T) - o(1)$, we are in the 'near-critical' percolation region.
The function $r(p)$ seems not so well understood for general graphs in this region...
But for the triangular lattice $T$, celebrated results of [Smirnov,Werner 2001] can be used to show

$$r(p) \to c \cdot (p - 1/2)^{-91/36+o(1)} \text{ as } p \to 1/2,$$

which implies that we can take $\ell = O(n^{91/36+\varepsilon} \cdot (n + \log(\delta^{-1}))$ for error probability $\delta$.

# Step 3: $t$-Reliable $n$-Colourings – Probabilistic Construction

In summary, we proved:

**Theorem**

*For any $\delta > 0$, we can construct a black-box protocol $\prod$ for $f_G$ such that*

- *If $t < n/2$, $\prod$ has communication complexity $O(n^{6.056}(n + \log \delta^{-1})^2)$ group elements.*
- *If $t \leq n/(2 + \epsilon)$ for some constant $\epsilon > 0$, $\prod$ has communication complexity $O(n\,(n + \log \delta^{-1})^2)$ group elements,*

*and the probability that $\prod$ is not $t$-private is at most $\delta$.*

# Extension: computing arbitrary functions

Our protocols easily generalize from computing $f_G(x_1, \ldots, x_n)$ to compute any G-circuit with two types of gates:

1. Mult: $(x, y) \mapsto x \cdot y$.
2. $\text{CMult}_{\alpha,\beta}$: $x \mapsto \alpha \cdot x \cdot \beta$

**Question:** Can any Boolean circuit be computed by a $G$-circuit, for some finite group $G$?

- Let $\phi_\sigma : \{0, 1\} \to G$ denote an encoding function mapping $0 \mapsto 1_G$ and $1 \mapsto \sigma$.
- $G$-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 $(x_1, \ldots, x_n) \in \{0, 1\}^n$.

# Extension: computing arbitrary functions

Our protocols easily generalize from computing $f_G(x_1, \ldots, x_n)$ to compute any G-circuit with two types of gates:

1. Mult: $(x, y) \mapsto x \cdot y$.
2. $\mathrm{CMult}_{\alpha,\beta}$: $x \mapsto \alpha \cdot x \cdot \beta$

**Question:** Can any Boolean circuit be computed by a $G$-circuit, for some finite group $G$?

- Let $\phi_\sigma : \{0, 1\} \to G$ denote an encoding function mapping $0 \mapsto 1_G$ and $1 \mapsto \sigma$.
- $G$-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 $(x_1, \ldots, x_n) \in \{0, 1\}^n$.

# Extension: computing arbitrary functions

Our protocols easily generalize from computing $f_G(x_1, \ldots, x_n)$ to compute any $G$-circuit with two types of gates:

1. Mult: $(x, y) \mapsto x \cdot y$.
2. $\text{CMult}_{\alpha,\beta}$: $x \mapsto \alpha \cdot x \cdot \beta$

**Question:** Can any Boolean circuit be computed by a $G$-circuit, for some finite group $G$?

- Let $\phi_\sigma : \{0, 1\} \to G$ denote an encoding function mapping $0 \mapsto 1_G$ and $1 \mapsto \sigma$.
- $G$-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 $(x_1, \ldots, x_n) \in \{0, 1\}^n$.

# Extension: computing arbitrary functions

**Theorem (Adapted from Barrington'86)**

*Let $C$ be a Boolean circuit consisting of $N_A$ 2-input AND gates, $N_N$ NOT gates. 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.*

Proof idea:

- Take encoding $\phi_\sigma$ mapping 0 to $1_{S_5}$ and 1 to $\sigma = (12345)$.
- Recall: $x, y \in S_5$ are conjugates if $x = h \cdot y \cdot h^{-1}$ for some $h \in S_5$.
- **Facts.**:
  - Set $J$ of all 5-cycles of $S_5$ is a conjugacy class of $S_5$.
  - $J$ contains two elements $\sigma_1, \sigma_2$ whose commutator $\sigma_1 \sigma_2 \sigma_1^{-1} \sigma_2^{-1}$ belongs to $J$.

# Extension: computing arbitrary functions

- Hence, for $\sigma, \sigma' \in J$, can convert an encoding $\phi_\sigma(x)$ w.r.t. $\sigma'$ to encoding $\phi_{\sigma'}(x)$ w.r.t. $\sigma'$ by a CMult gate:

$$x_{\sigma'} = h_{\sigma,\sigma'} \cdot x_\sigma \cdot h_{\sigma,\sigma'}^{-1}$$

- To compute AND $z = AND(x, y)$ w.r.t. encoding $\phi_{\sigma_1}$, given inputs $x_{\sigma_1}, y_{\sigma_1} \in S_5$:
  - Compute by encoding conversion $x_{\sigma_1^{-1}}$, $y_{\sigma_2}$, $y_{\sigma_2^{-1}}$.
  - Compute $z_c = x_{\sigma_1} y_{\sigma_2} x_{\sigma_1^{-1}} y_{\sigma_2^{-1}}$ ($z_c = [x_{\sigma_1}, y_{\sigma_2}]$ is an encoding of $z = AND(x, y)$ w.r.t. $c = [\sigma_1, \sigma_2]$).
  - Compute by encoding conversion $z_{\sigma_1}$.

# Extension: Security against active attacks

- Recently [SCN'12, to appear], we constructed variants of these protocols with active security
- Works for $t < n/3$ (optimal for active attacks)
- But, so far we can only make this work for graphs with $\ell$ exponential in $n$...

## Extension: Security against active attacks

- Recently [SCN'12, to appear], we constructed variants of these protocols with active security
- Works for $t < n/3$ (optimal for active attacks)
- But, so far we can only make this work for graphs with $\ell$ exponential in $n$...

# Extension: Security against active attacks

- Recently [SCN'12, to appear], we constructed variants of these protocols with active security
- Works for $t < n/3$ (optimal for active attacks)
- But, so far we can only make this work for graphs with $\ell$ exponential in $n$...

# Extension: Security against active attacks

Main ideas:

- Use a variant of the deterministic coloring, but with $2t+1$-subsets colouring the edges
- At each node, two incoming $2t+1$-subsets jointly perform the node multiplication and resharing:
  - All parties in intersection of incoming $2t+1$-subsets perform the multiplication; one is honest.
  - Consistency among products is verified by the honest majority in each $2t+1$-subset
- Problem in reducing exponential complexity:
  - Each $2t+1$-subset 'color' excludes a unique $t$-subset
  - Corresponding edge can only be used for one $I$-avoiding path
  - But in a $\mathcal{P}oly(n)$-sized graph, edges must be re-used for exp. many $I$'s!

# Extension: Security against active attacks

Main ideas:

- Use a variant of the deterministic coloring, but with $2t+1$-subsets colouring the edges
- At each node, two incoming $2t+1$-subsets jointly perform the node multiplication and resharing:
  - All parties in intersection of incoming $2t+1$-subsets perform the multiplication; one is honest.
  - Consistency among products is verified by the honest majority in each $2t+1$-subset
- Problem in reducing exponential complexity:
  - Each $2t+1$-subset 'color' excludes a unique $t$-subset
  - Corresponding edge can only be used for one $I$-avoiding path
  - But in a $\mathcal{P}oly(n)$-sized graph, edges must be re-used for exp. many $I$'s!

# Extension: Security against active attacks

Main ideas:

- Use a variant of the deterministic coloring, but with $2t+1$-subsets colouring the edges
- At each node, two incoming $2t+1$-subsets jointly perform the node multiplication and resharing:
  - All parties in intersection of incoming $2t+1$-subsets perform the multiplication; one is honest.
  - Consistency among products is verified by the honest majority in each $2t+1$-subset
- Problem in reducing exponential complexity:
  - Each $2t+1$-subset 'color' excludes a unique $t$-subset
  - Corresponding edge can only be used for one $I$-avoiding path
  - But in a $\mathcal{P}oly(n)$-sized graph, edges must be re-used for exp. many $I$'s!

# Extension: Security against active attacks

Main ideas:

- Use a variant of the deterministic coloring, but with $2t+1$-subsets colouring the edges
- At each node, two incoming $2t+1$-subsets jointly perform the node multiplication and resharing:
    - All parties in intersection of incoming $2t+1$-subsets perform the multiplication; one is honest.
    - Consistency among products is verified by the honest majority in each $2t+1$-subset
- Problem in reducing exponential complexity:
    - Each $2t+1$-subset 'color' excludes a unique $t$-subset
    - Corresponding edge can only be used for one $I$-avoiding path
    - But in a $\mathcal{P}oly(n)$-sized graph, edges must be re-used for exp. many $I$'s!

# Extension: Security against active attacks

Main ideas:

- Use a variant of the deterministic coloring, but with $2t+1$-subsets colouring the edges
- At each node, two incoming $2t+1$-subsets jointly perform the node multiplication and resharing:
  - All parties in intersection of incoming $2t+1$-subsets perform the multiplication; one is honest.
  - Consistency among products is verified by the honest majority in each $2t+1$-subset
- Problem in reducing exponential complexity:
  - Each $2t+1$-subset 'color' excludes a unique $t$-subset
  - Corresponding edge can only be used for one $I$-avoiding path
  - But in a $\mathcal{P}oly(n)$-sized graph, edges must be re-used for exp. many $I$'s!

# Conclusions and Open Problems

- We designed black-box $n$-Product protocols over any finite group based on $k$-of-$k$ secret sharing schemes by reduction to a combinatorial graph colouring problem
- Open Problems:
  - Can one obtain a deterministic construction of an admissible PDAG with $t$-reliable coloring, polynomial size, and optimal privacy ($t < n/2$)?
  - Can one obtain a protocol for black-box groups with active security having optimal resilience ($t < n/3$) and polynomial communication complexity?
  - Is it possible to construct black-box secure computation protocol for 'weaker' algebraic structures than groups?
  - Other applications for our protocols?