

# Efficiency Tradeoffs for Malicious Two-party Computation

Payman Mohassel    Matthew Franklin

Department of Computer Science  
University of California, Davis

PKC 2006

# Outline

- 1 Introduction
  - Secure Two-party Computation
  - Yao's Garbled Circuit Protocol
  - Motivation
- 2 Contributions
  - Overview
  - Protocols

# Outline

- 1 Introduction
  - Secure Two-party Computation
    - Yao's Garbled Circuit Protocol
    - Motivation
- 2 Contributions
  - Overview
  - Protocols

# Secure Two-party Computation

- Bob's input  $x$ , Alice's input  $y$ .
- $f(x, y) = (f_1(x, y), f_2(x, y))$ .
- Security:
  - Computationally bounded adversaries
  - *Semi-honest vs. malicious* adversaries

# Secure Two-party Computation

- Bob's input  $x$ , Alice's input  $y$ .
- $f(x, y) = (f_1(x, y), f_2(x, y))$ .
- Security:
  - Computationally bounded adversaries
  - *Semi-honest vs. malicious* adversaries

# Secure Two-party Computation

- Bob's input  $x$ , Alice's input  $y$ .
- $f(x, y) = (f_1(x, y), f_2(x, y))$ .
- Security:
  - Computationally bounded adversaries
  - *Semi-honest* vs. *malicious* adversaries

# Outline

- 1 Introduction
  - Secure Two-party Computation
  - Yao's Garbled Circuit Protocol
  - Motivation
- 2 Contributions
  - Overview
  - Protocols

# Yao's garbled circuit

- Circuit  $C$  computes  $f$ .
- Bob computes a garbled circuit:
- Random strings  $K_{j,0}$  and  $K_{j,1}$  for wire  $j$ .
- **Garbled truth table** for each gate.

Wire $i$	Wire $j$	Output wire $o$	Garbled output
$K_{i,0}$	$K_{j,0}$	$K_{o,0}$	$E_{K_{i,0}, K_{j,0}}(K_{o,0})$
$K_{i,0}$	$K_{j,1}$	$K_{o,1}$	$E_{K_{i,0}, K_{j,1}}(K_{o,1})$
$K_{i,1}$	$K_{j,0}$	$K_{o,1}$	$E_{K_{i,1}, K_{j,0}}(K_{o,1})$
$K_{i,1}$	$K_{j,1}$	$K_{o,1}$	$E_{K_{i,1}, K_{j,1}}(K_{o,1})$

- Translation table for output wires.

# Yao's garbled circuit

- Circuit  $C$  computes  $f$ .
- Bob computes a garbled circuit:
- Random strings  $K_{j,0}$  and  $K_{j,1}$  for wire  $j$ .
- **Garbled truth table** for each gate.

Wire $i$	Wire $j$	Output wire $o$	Garbled output
$K_{i,0}$	$K_{j,0}$	$K_{o,0}$	$E_{K_{i,0}, K_{j,0}}(K_{o,0})$
$K_{i,0}$	$K_{j,1}$	$K_{o,1}$	$E_{K_{i,0}, K_{j,1}}(K_{o,1})$
$K_{i,1}$	$K_{j,0}$	$K_{o,1}$	$E_{K_{i,1}, K_{j,0}}(K_{o,1})$
$K_{i,1}$	$K_{j,1}$	$K_{o,1}$	$E_{K_{i,1}, K_{j,1}}(K_{o,1})$

- Translation table for output wires.

# Yao's garbled circuit

- Circuit  $C$  computes  $f$ .
- Bob computes a garbled circuit:
- Random strings  $K_{j,0}$  and  $K_{j,1}$  for wire  $j$ .
- Garbled truth table for each gate.

Wire $i$	Wire $j$	Output wire $o$	Garbled output
$K_{i,0}$	$K_{j,0}$	$K_{o,0}$	$E_{K_{i,0}, K_{j,0}}(K_{o,0})$
$K_{i,0}$	$K_{j,1}$	$K_{o,1}$	$E_{K_{i,0}, K_{j,1}}(K_{o,1})$
$K_{i,1}$	$K_{j,0}$	$K_{o,1}$	$E_{K_{i,1}, K_{j,0}}(K_{o,1})$
$K_{i,1}$	$K_{j,1}$	$K_{o,1}$	$E_{K_{i,1}, K_{j,1}}(K_{o,1})$

- Translation table for output wires.

# Yao's garbled circuit

- Circuit  $C$  computes  $f$ .
- Bob computes a garbled circuit:
- Random strings  $K_{j,0}$  and  $K_{j,1}$  for wire  $j$ .
- **Garbled truth table** for each gate.

Wire $i$	Wire $j$	Output wire $o$	Garbled output
$K_{i,0}$	$K_{j,0}$	$K_{o,0}$	$E_{K_{i,0}, K_{j,0}}(K_{o,0})$
$K_{i,0}$	$K_{j,1}$	$K_{o,1}$	$E_{K_{i,0}, K_{j,1}}(K_{o,1})$
$K_{i,1}$	$K_{j,0}$	$K_{o,1}$	$E_{K_{i,1}, K_{j,0}}(K_{o,1})$
$K_{i,1}$	$K_{j,1}$	$K_{o,1}$	$E_{K_{i,1}, K_{j,1}}(K_{o,1})$

- Translation table for output wires.

# Yao's garbled circuit

- Circuit  $C$  computes  $f$ .
- Bob computes a garbled circuit:
- Random strings  $K_{j,0}$  and  $K_{j,1}$  for wire  $j$ .
- **Garbled truth table** for each gate.

Wire $i$	Wire $j$	Output wire $o$	Garbled output
$K_{i,0}$	$K_{j,0}$	$K_{o,0}$	$E_{K_{i,0}, K_{j,0}}(K_{o,0})$
$K_{i,0}$	$K_{j,1}$	$K_{o,1}$	$E_{K_{i,0}, K_{j,1}}(K_{o,1})$
$K_{i,1}$	$K_{j,0}$	$K_{o,1}$	$E_{K_{i,1}, K_{j,0}}(K_{o,1})$
$K_{i,1}$	$K_{j,1}$	$K_{o,1}$	$E_{K_{i,1}, K_{j,1}}(K_{o,1})$

- Translation table for output wires.

# Yao's protocol cont'd

- 1 Bob sends the garbled circuit to Alice.
- 2 Bob sends his garbled input to Alice.
- 3 1-out-of-2 *OT* for Alice's input wires.
  - Alice's input:  $a_i \in \{0, 1\}$
  - Bob's input:  $K_{i,0}$  and  $K_{i,1}$
  - Alice learns  $K_{i,a_i}$
- 4 Alice evaluates the circuit.

- $O(|C|)$  symmetric encryptions.
- $O(I)$  oblivious transfers.

# Yao's protocol cont'd

- 1 Bob sends the garbled circuit to Alice.
- 2 Bob sends his garbled input to Alice.
- 3 1-out-of-2 *OT* for Alice's input wires.
  - Alice's input:  $a_i \in \{0, 1\}$
  - Bob's input:  $K_{i,0}$  and  $K_{i,1}$
  - Alice learns  $K_{i,a_i}$
- 4 Alice evaluates the circuit.

- $O(|C|)$  symmetric encryptions.
- $O(I)$  oblivious transfers.

# Yao's protocol cont'd

- 1 Bob sends the garbled circuit to Alice.
- 2 Bob sends his garbled input to Alice.
- 3 1-out-of-2 OT for Alice's input wires.
  - Alice's input:  $a_i \in \{0, 1\}$
  - Bob's input:  $K_{i,0}$  and  $K_{i,1}$
  - Alice learns  $K_{i,a_i}$
- 4 Alice evaluates the circuit.

- $O(|C|)$  symmetric encryptions.
- $O(I)$  oblivious transfers.

# Yao's protocol cont'd

- 1 Bob sends the garbled circuit to Alice.
- 2 Bob sends his garbled input to Alice.
- 3 1-out-of-2 *OT* for Alice's input wires.
  - Alice's input:  $a_i \in \{0, 1\}$
  - Bob's input:  $K_{i,0}$  and  $K_{i,1}$
  - Alice learns  $K_{i,a_i}$
- 4 Alice evaluates the circuit.

- $O(|C|)$  symmetric encryptions.
- $O(I)$  oblivious transfers.

# Yao's protocol cont'd

- 1 Bob sends the garbled circuit to Alice.
- 2 Bob sends his garbled input to Alice.
- 3 1-out-of-2 OT for Alice's input wires.
  - Alice's input:  $a_i \in \{0, 1\}$
  - Bob's input:  $K_{i,0}$  and  $K_{i,1}$
  - Alice learns  $K_{i,a_i}$
- 4 Alice evaluates the circuit.

- $O(|C|)$  symmetric encryptions.
- $O(I)$  oblivious transfers.

# Yao's protocol cont'd

- 1 Bob sends the garbled circuit to Alice.
- 2 Bob sends his garbled input to Alice.
- 3 1-out-of-2 OT for Alice's input wires.
  - Alice's input:  $a_i \in \{0, 1\}$
  - Bob's input:  $K_{i,0}$  and  $K_{i,1}$
  - Alice learns  $K_{i,a_i}$
- 4 Alice evaluates the circuit.

- $O(|C|)$  symmetric encryptions.
- $O(I)$  oblivious transfers.

# Yao's protocol cont'd

- 1 Bob sends the garbled circuit to Alice.
- 2 Bob sends his garbled input to Alice.
- 3 1-out-of-2 *OT* for Alice's input wires.
  - Alice's input:  $a_i \in \{0, 1\}$
  - Bob's input:  $K_{i,0}$  and  $K_{i,1}$
  - Alice learns  $K_{i,a_i}$
- 4 Alice evaluates the circuit.

- $O(|C|)$  symmetric encryptions.
- $O(I)$  oblivious transfers.

# Outline

- 1 Introduction
  - Secure Two-party Computation
  - Yao's Garbled Circuit Protocol
  - Motivation
- 2 Contributions
  - Overview
  - Protocols

# Motivation

- Yao's protocol can be practical:
  - Efficient for functions with **small** circuits.
  - Used to implement parts of some protocols more efficiently.
  - *Fairplay*: Implementation of Yao's, [MNPS04].

Efficient protocols for the **malicious** case needed!

# Motivation

- Yao's protocol can be practical:
  - Efficient for functions with **small** circuits.
  - Used to implement parts of some protocols more efficiently.
  - *Fairplay*: Implementation of Yao's, [MNPS04].

Efficient protocols for the **malicious** case needed!

# Motivation

- Yao's protocol can be practical:
  - Efficient for functions with **small** circuits.
  - Used to implement parts of some protocols more efficiently.
  - *Fairplay*: Implementation of Yao's, [MNPS04].

Efficient protocols for the **malicious** case needed!

# Motivation

- Yao's protocol can be practical:
  - Efficient for functions with **small** circuits.
  - Used to implement parts of some protocols more efficiently.
  - *Fairplay*: Implementation of Yao's, [MNPS04].

Efficient protocols for the **malicious** case needed!

# Malicious Adversaries

- Malicious behaviors:
  - Sending an incorrect garbled circuit.
  - Sending wrong inputs during the OT's.
- General **zero-knowledge** proofs [GMW86, 87].
  - Inverse exponential cheating probability
  - Inefficient
- **cut-and-choose** techniques.
  - Fairplay, [MNPS04]
  - Cheating probability not inverse exponential
  - Efficient

# Malicious Adversaries

- Malicious behaviors:
  - Sending an incorrect garbled circuit.
  - Sending wrong inputs during the OT's.
- General **zero-knowledge** proofs [GMW86, 87].
  - Inverse exponential cheating probability
  - Inefficient
- **cut-and-choose** techniques.
  - Fairplay, [MNPS04]
  - Cheating probability not inverse exponential
  - Efficient

# Malicious Adversaries

- Malicious behaviors:
  - Sending an incorrect garbled circuit.
  - Sending wrong inputs during the OT's.
- General **zero-knowledge** proofs [GMW86, 87].
  - Inverse exponential cheating probability
  - Inefficient
- **cut-and-choose** techniques.
  - Fairplay, [MNPS04]
  - Cheating probability not inverse exponential
  - Efficient

# Malicious Adversaries

- Malicious behaviors:
  - Sending an incorrect garbled circuit.
  - Sending wrong inputs during the OT's.
- General **zero-knowledge** proofs [GMW86, 87].
  - Inverse exponential cheating probability
  - Inefficient
- **cut-and-choose** techniques.
  - Fairplay, [MNPS04]
  - Cheating probability not inverse exponential
  - Efficient

# Malicious Adversaries

- Malicious behaviors:
  - Sending an incorrect garbled circuit.
  - Sending wrong inputs during the OT's.
- General **zero-knowledge** proofs [GMW86, 87].
  - Inverse exponential cheating probability
  - Inefficient
- **cut-and-choose** techniques.
  - Fairplay, [MNPS04]
  - Cheating probability not inverse exponential
  - Efficient

# Malicious Adversaries

- Malicious behaviors:
  - Sending an incorrect garbled circuit.
  - Sending wrong inputs during the OT's.
- General **zero-knowledge** proofs [GMW86, 87].
  - Inverse exponential cheating probability
  - Inefficient
- **cut-and-choose** techniques.
  - Fairplay, [MNPS04]
  - Cheating probability not inverse exponential
  - Efficient

# Outline

- 1 Introduction
  - Secure Two-party Computation
  - Yao's Garbled Circuit Protocol
  - Motivation
- 2 Contributions
  - Overview
  - Protocols

# Contributions

- Secure protocols for the malicious case.
  - Inverse exponential cheating probability
  - Efficient
- A more careful look at cut-and-choose techniques.
- Suggest tradeoffs between *security* and *efficiency*.
- Note: our protocols are secure when only Alice learns the output.

# Contributions

- Secure protocols for the malicious case.
  - Inverse exponential cheating probability
  - Efficient
- A more careful look at cut-and-choose techniques.
- Suggest tradeoffs between *security* and *efficiency*.
- Note: our protocols are secure when only Alice learns the output.

# Contributions

- Secure protocols for the malicious case.
  - Inverse exponential cheating probability
  - Efficient
- A more careful look at cut-and-choose techniques.
- Suggest tradeoffs between *security* and *efficiency*.
- Note: our protocols are secure when only Alice learns the output.

# Contributions

- Secure protocols for the malicious case.
  - Inverse exponential cheating probability
  - Efficient
- A more careful look at cut-and-choose techniques.
- Suggest tradeoffs between *security* and *efficiency*.
- Note: our protocols are secure when only Alice learns the output.

# Outline

- 1 Introduction
  - Secure Two-party Computation
  - Yao's Garbled Circuit Protocol
  - Motivation
  
- 2 Contributions
  - Overview
  - Protocols

# Cut-and-choose techniques

- 1 Bob sends  $m$  garbled circuits.
  - 2 Alice randomly checks  $i$  circuits for correctness.
  - 3 Alice evaluates the  $m - i$  remaining circuits and outputs the majority.
- Bob sends his garbled inputs for the  $m - i$  circuits.
  - Bob proves that he sent the same input to those circuits.
  - Alice learns her garbled inputs through the oblivious transfers
  - Alice verifies that she received garbled form of her input values.
  - If  $i = m - 1$ , cheating probability  $1/m$ .
  - If  $i = m/2$ , cheating probability exponentially small in  $m$ .

# Cut-and-choose techniques

- 1 Bob sends  $m$  garbled circuits.
  - 2 Alice randomly checks  $i$  circuits for correctness.
  - 3 Alice evaluates the  $m - i$  remaining circuits and outputs the majority.
- Bob sends his garbled inputs for the  $m - i$  circuits.
  - Bob proves that he sent the same input to those circuits.
  - Alice learns her garbled inputs through the oblivious transfers
  - Alice verifies that she received garbled form of her input values.
  - If  $i = m - 1$ , cheating probability  $1/m$ .
  - If  $i = m/2$ , cheating probability exponentially small in  $m$ .

# Cut-and-choose techniques

- 1 Bob sends  $m$  garbled circuits.
  - 2 Alice randomly checks  $i$  circuits for correctness.
  - 3 Alice evaluates the  $m - i$  remaining circuits and outputs the majority.
- Bob sends his garbled inputs for the  $m - i$  circuits.
  - Bob proves that he sent the same input to those circuits.
  - Alice learns her garbled inputs through the oblivious transfers
  - Alice verifies that she received garbled form of her input values.
  - If  $i = m - 1$ , cheating probability  $1/m$ .
  - If  $i = m/2$ , cheating probability exponentially small in  $m$ .

# Cut-and-choose techniques

- 1 Bob sends  $m$  garbled circuits.
  - 2 Alice randomly checks  $i$  circuits for correctness.
  - 3 Alice evaluates the  $m - i$  remaining circuits and outputs the majority.
- Bob sends his garbled inputs for the  $m - i$  circuits.
  - **Bob proves that he sent the same input to those circuits.**
  - Alice learns her garbled inputs through the oblivious transfers
  - **Alice verifies that she received garbled form of her input values.**
  - If  $i = m - 1$ , cheating probability  $1/m$ .
  - If  $i = m/2$ , cheating probability exponentially small in  $m$ .

# Cut-and-choose techniques

- 1 Bob sends  $m$  garbled circuits.
  - 2 Alice randomly checks  $i$  circuits for correctness.
  - 3 Alice evaluates the  $m - i$  remaining circuits and outputs the majority.
- Bob sends his garbled inputs for the  $m - i$  circuits.
  - **Bob proves that he sent the same input to those circuits.**
  - Alice learns her garbled inputs through the oblivious transfers
  - **Alice verifies that she received garbled form of her input values.**
  - If  $i = m - 1$ , cheating probability  $1/m$ .
  - If  $i = m/2$ , cheating probability exponentially small in  $m$ .

# Cut-and-choose techniques

- 1 Bob sends  $m$  garbled circuits.
  - 2 Alice randomly checks  $i$  circuits for correctness.
  - 3 Alice evaluates the  $m - i$  remaining circuits and outputs the majority.
- Bob sends his garbled inputs for the  $m - i$  circuits.
  - **Bob proves that he sent the same input to those circuits.**
  - Alice learns her garbled inputs through the oblivious transfers
  - **Alice verifies that she received garbled form of her input values.**
  - If  $i = m - 1$ , cheating probability  $1/m$ .
  - If  $i = m/2$ , cheating probability exponentially small in  $m$ .

# Cut-and-choose techniques

- 1 Bob sends  $m$  garbled circuits.
  - 2 Alice randomly checks  $i$  circuits for correctness.
  - 3 Alice evaluates the  $m - i$  remaining circuits and outputs the majority.
- Bob sends his garbled inputs for the  $m - i$  circuits.
  - Bob proves that he sent the same input to those circuits.
  - Alice learns her garbled inputs through the oblivious transfers
  - Alice verifies that she received garbled form of her input values.
  - If  $i = m - 1$ , cheating probability  $1/m$ .
  - If  $i = m/2$ , cheating probability exponentially small in  $m$ .

# Our schemes

Alice verifies that she received garbled form of her input values.

- Bob includes commitment  $z_{j,i,b}$  of the tuple  $(j, i, b, K_{i,b}^{(j)})$ .
- Witness for decommitment  $w_{j,i,b}$ .
- For circuit  $j \in [1..m]$ ,  $b \in \{0, 1\}$ , Alice's wire  $i$ .
- Bob opens all the commitments for the revealed circuits.
- Bob obviously transfers  $(K_{i,0}^{(j)}, w_{j,i,0})$  or  $(K_{i,1}^{(j)}, w_{j,i,1})$  for the remaining circuits.

# Our schemes

Alice verifies that she received garbled form of her input values.

- Bob includes commitment  $z_{j,i,b}$  of the tuple  $(j, i, b, K_{i,b}^{(j)})$ .
- Witness for decommitment  $w_{j,i,b}$ .
- For circuit  $j \in [1..m]$ ,  $b \in \{0, 1\}$ , Alice's wire  $i$ .
- Bob opens all the commitments for the revealed circuits.
- Bob obviously transfers  $(K_{i,0}^{(j)}, w_{j,i,0})$  or  $(K_{i,1}^{(j)}, w_{j,i,1})$  for the remaining circuits.

# Our schemes

Alice verifies that she received garbled form of her input values.

- Bob includes commitment  $z_{j,i,b}$  of the tuple  $(j, i, b, K_{i,b}^{(j)})$ .
- Witness for decommitment  $w_{j,i,b}$ .
- For circuit  $j \in [1..m]$ ,  $b \in \{0, 1\}$ , Alice's wire  $i$ .
- Bob opens all the commitments for the revealed circuits.
- Bob obviously transfers  $(K_{i,0}^{(j)}, w_{j,i,0})$  or  $(K_{i,1}^{(j)}, w_{j,i,1})$  for the remaining circuits.

# Our schemes

Alice verifies that she received garbled form of her input values.

- Bob includes commitment  $z_{j,i,b}$  of the tuple  $(j, i, b, K_{i,b}^{(j)})$ .
- Witness for decommitment  $w_{j,i,b}$ .
- For circuit  $j \in [1..m]$ ,  $b \in \{0, 1\}$ , Alice's wire  $i$ .
- Bob opens all the commitments for the revealed circuits.
- Bob obviously transfers  $(K_{i,0}^{(j)}, w_{j,i,0})$  or  $(K_{i,1}^{(j)}, w_{j,i,1})$  for the remaining circuits.

# Committed-input scheme

Bob proves that he sent the same input to the remaining circuits.

- A commitment scheme with proof of equality.
- e.g., Pederson's commitment scheme, [Ped91]:
  - Bob sends  $(x, r) = (x, H(x, r))$  to Alice who publishes  $(x, r)$ .
  - Bob sends the commitments for the remaining circuits.
  - Bob sends his garbled input for the remaining circuits.
  - Alice proves their equality.
- $O(ml)$  extra exponentiations.

# Committed-input scheme

Bob proves that he sent the same input to the remaining circuits.

- A commitment scheme with proof of equality.
- e.g., Pederson's commitment scheme, [Ped91]:
  - Bob sends  $K_{i,b}^{(j)} = g^b h^{r_{i,b}^{(j)}}$ , random  $r_{i,b}^{(j)}$ , for Bob's wire  $i$ .
  - Bob opens the commitments for the revealed circuits.
  - Bob sends his garbled input for the remaining circuits.
  - Bob proves their equality.
- $O(ml)$  extra exponentiations.

# Committed-input scheme

Bob proves that he sent the same input to the remaining circuits.

- A commitment scheme with proof of equality.
- e.g., Pederson's commitment scheme, [Ped91]:
  - Bob sends  $K_{i,b}^{(j)} = g^b h^{r_{i,b}^{(j)}}$ , random  $r_{i,b}^{(j)}$ , for Bob's wire  $i$ .
  - Bob opens the commitments for the revealed circuits.
  - Bob sends his garbled input for the remaining circuits.
  - Bob proves their equality.
- $O(ml)$  extra exponentiations.

# Committed-input scheme

Bob proves that he sent the same input to the remaining circuits.

- A commitment scheme with proof of equality.
- e.g., Pederson's commitment scheme, [Ped91]:
  - Bob sends  $K_{i,b}^{(j)} = g^b h^{r_{i,b}^{(j)}}$ , random  $r_{i,b}^{(j)}$ , for Bob's wire  $i$ .
  - Bob opens the commitments for the revealed circuits.
    - Bob sends his garbled input for the remaining circuits.
    - Bob proves their equality.
- $O(ml)$  extra exponentiations.

# Committed-input scheme

Bob proves that he sent the same input to the remaining circuits.

- A commitment scheme with proof of equality.
- e.g., Pederson's commitment scheme, [Ped91]:
  - Bob sends  $K_{i,b}^{(j)} = g^b h^{r_{i,b}^{(j)}}$ , random  $r_{i,b}^{(j)}$ , for Bob's wire  $i$ .
  - Bob opens the commitments for the revealed circuits.
  - Bob sends his garbled input for the remaining circuits.
    - Bob proves their equality.
- $O(ml)$  extra exponentiations.

# Committed-input scheme

Bob proves that he sent the same input to the remaining circuits.

- A commitment scheme with proof of equality.
- e.g., Pederson's commitment scheme, [Ped91]:
  - Bob sends  $K_{i,b}^{(j)} = g^b h^{r_{i,b}^{(j)}}$ , random  $r_{i,b}^{(j)}$ , for Bob's wire  $i$ .
  - Bob opens the commitments for the revealed circuits.
  - Bob sends his garbled input for the remaining circuits.
  - Bob proves their equality.
- $O(ml)$  extra exponentiations.

# Committed-input scheme

Bob proves that he sent the same input to the remaining circuits.

- A commitment scheme with proof of equality.
- e.g., Pederson's commitment scheme, [Ped91]:
  - Bob sends  $K_{i,b}^{(j)} = g^b h^{r_{i,b}^{(j)}}$ , random  $r_{i,b}^{(j)}$ , for Bob's wire  $i$ .
  - Bob opens the commitments for the revealed circuits.
  - Bob sends his garbled input for the remaining circuits.
  - Bob proves their equality.
- $O(ml)$  extra exponentiations.

# Equality-checker scheme

**Bob proves that he sent the same input to the remaining circuits.**

- Equality-checker for  $C_f$  and  $C_g$ .

• Bob's commitment to the value  $x$  is  $(C_f(x), C_g(x))$ .  
• For  $C_f$  and  $C_g$  Bob sends  $(C_f(x), C_g(x))$ .  
• Bob's decommitment key is  $(x, C_f(x), C_g(x))$ .  
• Bob reveals  $x$  by sending  $(x, C_f(x), C_g(x))$  to the state inspectors in the  $C_f$  circuit.  
• Bob reveals  $x$  by sending  $(x, C_f(x), C_g(x))$  to the state inspectors in the  $C_g$  circuit.  
• Alice verifies correctness of revealed equality checker.  
• Alice uses the rest of the equality checker to verify remaining circuits.

- No additional exponentiations.

# Equality-checker scheme

Bob proves that he sent the same input to the remaining circuits.

- **Equality-checker** for  $C_j$  and  $C_{j'}$ .
  - $Z_{j,j',i,b}$ , Bob's commitment to the tuple  $(j, j', K_{i,b}^{(j)}, K_{i,b}^{(j')})$ .
  - For  $1 \leq j < j' \leq m$ , Bob's wires  $i$ , and  $b \in \{0, 1\}$ .
  - Binds random key strings corresponding to same bit value  $b$  for the same input wire, in two different circuits.
  - Bob sends  $m(m-1)/2$  equality-checkers to Alice.
  - Alice verifies correctness of revealed equality checkers.
  - Alice uses the rest to verify that Bob sent the same input to remaining circuits.
- No additional exponentiations.

# Equality-checker scheme

Bob proves that he sent the same input to the remaining circuits.

- **Equality-checker** for  $C_j$  and  $C_{j'}$ .
  - $Z_{j,j',i,b}$ , Bob's commitment to the tuple  $(j, j', K_{i,b}^{(j)}, K_{i,b}^{(j')})$ .
  - For  $1 \leq j < j' \leq m$ , Bob's wires  $i$ , and  $b \in \{0, 1\}$ .
  - Binds random key strings corresponding to same bit value  $b$  for the same input wire, in two different circuits.
  - Bob sends  $m(m-1)/2$  equality-checkers to Alice.
  - Alice verifies correctness of revealed equality checkers.
  - Alice uses the rest to verify that Bob sent the same input to remaining circuits.
- No additional exponentiations.

# Equality-checker scheme

Bob proves that he sent the same input to the remaining circuits.

- **Equality-checker** for  $C_j$  and  $C_{j'}$ .
  - $Z_{j,j',i,b}$ , Bob's commitment to the tuple  $(j, j', K_{i,b}^{(j)}, K_{i,b}^{(j')})$ .
  - For  $1 \leq j < j' \leq m$ , Bob's wires  $i$ , and  $b \in \{0, 1\}$ .
  - Binds random key strings corresponding to same bit value  $b$  for the same input wire, in two different circuits.
  - Bob sends  $m(m-1)/2$  equality-checkers to Alice.
  - Alice verifies correctness of revealed equality checkers.
  - Alice uses the rest to verify that Bob sent the same input to remaining circuits.
- No additional exponentiations.

# Equality-checker scheme

Bob proves that he sent the same input to the remaining circuits.

- **Equality-checker** for  $C_j$  and  $C_{j'}$ .
  - $Z_{j,j',i,b}$ , Bob's commitment to the tuple  $(j, j', K_{i,b}^{(j)}, K_{i,b}^{(j')})$ .
  - For  $1 \leq j < j' \leq m$ , Bob's wires  $i$ , and  $b \in \{0, 1\}$ .
  - Binds random key strings corresponding to same bit value  $b$  for the same input wire, in two different circuits.
  - Bob sends  $m(m-1)/2$  equality-checkers to Alice.
  - Alice verifies correctness of revealed equality checkers.
    - Alice uses the rest to verify that Bob sent the same input to remaining circuits.
- No additional exponentiations.

# Equality-checker scheme

Bob proves that he sent the same input to the remaining circuits.

- **Equality-checker** for  $C_j$  and  $C_{j'}$ .
  - $z_{j,j',i,b}$ , Bob's commitment to the tuple  $(j, j', K_{i,b}^{(j)}, K_{i,b}^{(j')})$ .
  - For  $1 \leq j < j' \leq m$ , Bob's wires  $i$ , and  $b \in \{0, 1\}$ .
  - Binds random key strings corresponding to same bit value  $b$  for the same input wire, in two different circuits.
  - Bob sends  $m(m-1)/2$  equality-checkers to Alice.
  - Alice verifies correctness of revealed equality checkers.
  - Alice uses the rest to verify that Bob sent the same input to remaining circuits.
- No additional exponentiations.

# Equality-checker scheme

Bob proves that he sent the same input to the remaining circuits.

- **Equality-checker** for  $C_j$  and  $C_{j'}$ .
  - $Z_{j,j',i,b}$ , Bob's commitment to the tuple  $(j, j', K_{i,b}^{(j)}, K_{i,b}^{(j')})$ .
  - For  $1 \leq j < j' \leq m$ , Bob's wires  $i$ , and  $b \in \{0, 1\}$ .
  - Binds random key strings corresponding to same bit value  $b$  for the same input wire, in two different circuits.
  - Bob sends  $m(m-1)/2$  equality-checkers to Alice.
  - Alice verifies correctness of revealed equality checkers.
  - Alice uses the rest to verify that Bob sent the same input to remaining circuits.
- No additional exponentiations.

# Leaking information

- **1-leaked model**
- Malicious Bob learns  $f(x_{bob}, x_{alice})$  and *one extra bit*, Alice learns  $f(y_{bob}, x_{alice})$  (where Bob chooses  $x_{bob}, y_{bob}$ ).
- We give more efficient protocols secure in this model.
- Please see the paper for more detail.

# Leaking information

- **1-leaked model**
- Malicious Bob learns  $f(x_{bob}, x_{alice})$  and *one extra bit*, Alice learns  $f(y_{bob}, x_{alice})$  (where Bob chooses  $x_{bob}, y_{bob}$ ).
- We give more efficient protocols secure in this model.
- Please see the paper for more detail.

# Leaking information

- **1-leaked model**
- Malicious Bob learns  $f(x_{bob}, x_{alice})$  and *one extra bit*, Alice learns  $f(y_{bob}, x_{alice})$  (where Bob chooses  $x_{bob}, y_{bob}$ ).
- We give more efficient protocols secure in this model.
- Please see the paper for more detail.

# Leaking information

- **1-leaked model**
- Malicious Bob learns  $f(x_{bob}, x_{alice})$  and *one extra bit*, Alice learns  $f(y_{bob}, x_{alice})$  (where Bob chooses  $x_{bob}, y_{bob}$ ).
- We give more efficient protocols secure in this model.
- Please see the paper for more detail.

# Summary

- **Secure protocols for the malicious case.**
- More efficient than general zero-knowledge.
- A more careful look at cut-and-choose techniques.
- Tradeoffs between *security* and *efficiency*.
- There is more work to be done in this area!
  - Efficient protocols where both parties learn the output
  - Efficient protocols in alternative leakage models

# Summary

- Secure protocols for the malicious case.
- More efficient than general zero-knowledge.
- A more careful look at cut-and-choose techniques.
- Tradeoffs between *security* and *efficiency*.
- There is more work to be done in this area!
  - Efficient protocols where both parties learn the output
  - Efficient protocols in alternative leakage models

# Summary

- Secure protocols for the malicious case.
- More efficient than general zero-knowledge.
- A more careful look at cut-and-choose techniques.
- Tradeoffs between *security* and *efficiency*.
- There is more work to be done in this area!
  - Efficient protocols where both parties learn the output
  - Efficient protocols in alternative leakage models

# Summary

- Secure protocols for the malicious case.
- More efficient than general zero-knowledge.
- A more careful look at cut-and-choose techniques.
- Tradeoffs between *security* and *efficiency*.
- There is more work to be done in this area!
  - Efficient protocols where both parties learn the output
  - Efficient protocols in alternative leakage models

# Summary

- Secure protocols for the malicious case.
- More efficient than general zero-knowledge.
- A more careful look at cut-and-choose techniques.
- Tradeoffs between *security* and *efficiency*.
- There is more work to be done in this area!
  - Efficient protocols where both parties learn the output
  - Efficient protocols in alternative leakage models

# Summary

- Secure protocols for the malicious case.
- More efficient than general zero-knowledge.
- A more careful look at cut-and-choose techniques.
- Tradeoffs between *security* and *efficiency*.
- There is more work to be done in this area!
  - Efficient protocols where both parties learn the output
  - Efficient protocols in alternative leakage models

# Summary

- Secure protocols for the malicious case.
- More efficient than general zero-knowledge.
- A more careful look at cut-and-choose techniques.
- Tradeoffs between *security* and *efficiency*.
- There is more work to be done in this area!
  - Efficient protocols where both parties learn the output
  - Efficient protocols in alternative leakage models

# Questions

Thank you!  
Questions?



# Concrete results

- Fairplay scheme

Error probability	And	Billionaires	PIR	Median
$\frac{1}{100}$	$(176 * 10^2, 16)$	$(1092 * 10^2, 64)$	$(4976 * 10^2, 12)$	$(17916 * 10^2, 320)$
$\frac{1}{1000}$	$(176 * 10^3, 16)$	$(1092 * 10^3, 64)$	$(4976 * 10^3, 12)$	$(17916 * 10^3, 320)$
$\frac{1}{10000}$	$(176 * 10^4, 16)$	$(1092 * 10^4, 64)$	$(4976 * 10^4, 12)$	$(17916 * 10^4, 320)$
$\frac{1}{1000000}$	$(176 * 10^6, 16)$	$(1092 * 10^6, 64)$	$(4976 * 10^6, 12)$	$(17916 * 10^6, 320)$

- Committed-input scheme

Error probability	And	Billionaires	PIR	Median
$\frac{1}{100}$	(4677, 441)	(29020, 1764)	(132239, 25524)	(476125, 8824)
$\frac{1}{1000}$	(7015, 653)	(43530, 2615)	(198358, 38280)	(714187, 13076)
$\frac{1}{10000}$	(9354, 866)	(58040, 3465)	(264478, 51036)	(952250, 17328)
$\frac{1}{1000000}$	(14031, 1291)	(87061, 5166)	(396717, 76549)	(1428375, 25832)

- Equality-checker scheme

Error probability	And	Billionaires	PIR	Median
$\frac{1}{100}$	(19700, 16)	(94380, 64)	(961112, 12)	(968439, 320)
$\frac{1}{1000}$	(39100, 16)	(179708, 64)	(2013733, 12)	(1643347, 320)
$\frac{1}{10000}$	(64882, 16)	(290462, 64)	(3447731, 12)	(2445380, 320)
$\frac{1}{1000000}$	(135460, 16)	(588243, 64)	(7459858, 12)	(4430824, 320)

# Functions

- **AND**: bit-wise AND on two inputs of size 8; circuit has 32 gates
- **Billionaires**: Compares two 32-bit integers; circuit has 256 gates
- **PIR**: Bob's input size is 480 bits, Alice's input size is 6 bit; circuit has 1229 gates
- **Median**: Finds median of two sorted arrays, input for both Alice and Bob are ten 16-bit numbers, circuit size is 4383 gates