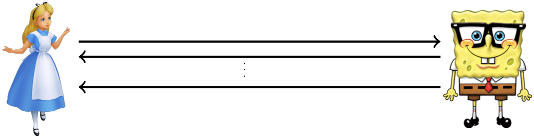
Non-Interactive Secure Computation Based on Cut-and-Choose

Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva

May 14, 2014

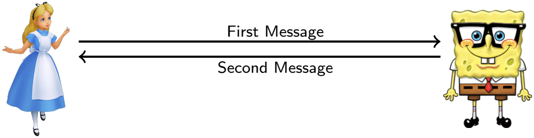
The Big Picture

- Secure Two Party Computation (2PC)
  - In presence of *malicious/active* adversaries.



The Big Picture

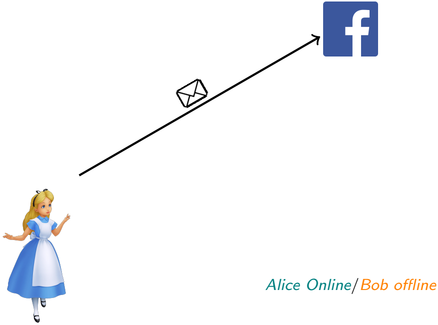
- Secure Two Party Computation (2PC)
  - In presence of *malicious/active* adversaries.
- Non-interactive computation
  - A *single* message is sent and a *single* message received.



The Big Picture

- Secure Two Party Computation (2PC)
  - In presence of *malicious/active* adversaries.
- Non-interactive computation
  - A *single* message is sent and a *single* message received.
- Practical
  - $\sim$ 6.4 seconds for AES circuit evaluation on a common laptop.

Motivation



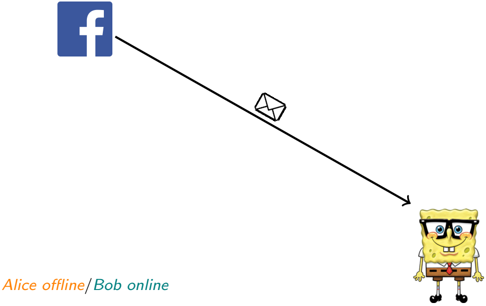
- General private computation.
- Asynchronous communications (i.e. email).
- Example usecases:
    - Private Set Intersection
    - DNA ancestry computation
    - ...

Motivation

Alice offline/*Bob online*

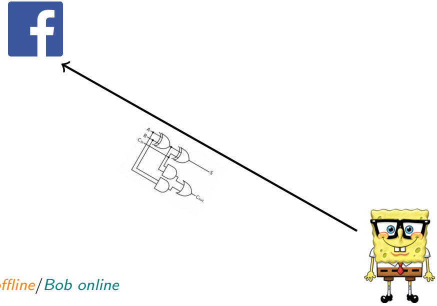
- General private computation.
- Asynchronous communications (i.e. email).
- Example usecases:
  - Private Set Intersection
  - DNA ancestry computation
  - ...

Motivation



- General private computation.
- Asynchronous communications (i.e. email).
- Example usecases:
    - Private Set Intersection
    - DNA ancestry computation
    - ...

Motivation



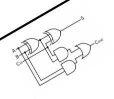
- General private computation.
- Asynchronous communications (i.e. email).
- Example usecases:
  - Private Set Intersection
  - DNA ancestry computation
  - ...

Motivation

Alice Online/*Bob offline*

- General private computation.
- Asynchronous communications (i.e. email).
- Example usecases:
  - Private Set Intersection
  - DNA ancestry computation
  - ...

Motivation



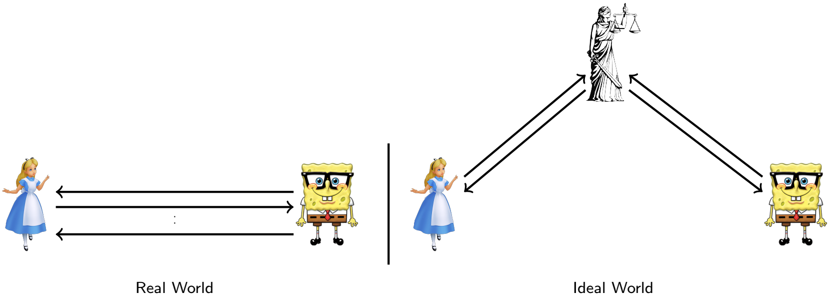
Alice Online/*Bob offline*

- General private computation.
- Asynchronous communications (i.e. email).
- Example usecases:
  - Private Set Intersection
  - DNA ancestry computation
  - ...

Outline

Security of 2PC

- Malicious adversary: might deviate from the protocol
- Computationally bounded adversary.
- Security is proved by simulation in Real/Ideal-world paradigm.
  - If an attack can be launched in Real-world,
  - then, Ideal-world will be compromised.



2PC Using Yao (Non-interactive)

- Semi-honest setting.
- *Easy to change* to *NISC*.

2PC Using Yao (Non-interactive)

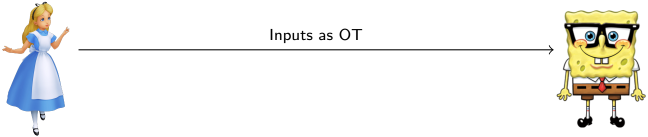
- Semi-honest setting.
- *Easy to change* to *NISC*.
- Fix terminology of the roles.

2PC Using Yao (Non-interactive)

- Semi-honest setting.
- *Easy to change* to *NISC*.
- Fix terminology of the roles.
- 2 message, universally composable OT [PVW08].

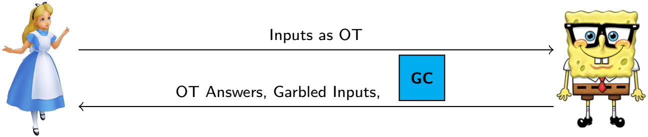
2PC Using Yao (Non-interactive)

- Semi-honest setting.
- *Easy to change* to *NISC*.
- Fix terminology of the roles.
- 2 message, universally composable OT [PVW08].
- First Message
  - Sends his inputs as OT inputs.



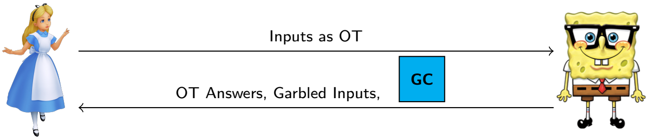
2PC Using Yao (Non-interactive)

- Semi-honest setting.
- *Easy to change* to *NISC*.
- Fix terminology of the roles.
- 2 message, universally composable OT [PVW08].
- First Message
  - Sends his inputs as OT inputs.
- Second Message
  - Sends his garbled inputs.
  - Sends the garbled circuit.
  - Sends OT answers corresponding to the Receiver's input.



2PC Using Yao (Non-interactive)

- Semi-honest setting.
- *Easy to change* to *NISC*.
- Fix terminology of the roles.
- 2 message, universally composable OT [PVW08].
- First Message
  - Sends his inputs as OT inputs.
- Second Message
  - Sends his garbled inputs.
  - Sends the garbled circuit.
  - Sends OT answers corresponding to the Receiver's input.
- Receiver
  - Evaluates the circuit.



Malicious 2PC Approaches (constant round approaches)

- *Not so easy!*

	Interactive	Non-interactive
Not Practical		
Asymptotically Efficient		
Highly Practical		

Malicious 2PC Approaches (constant round approaches)

- *Not so easy!*
- A spectrum of approaches!

	Interactive	Non-interactive
Not Practical	*Yao+ZKP*	*Yao+ZKP*
Asymptotically Efficient		
Highly Practical		

Malicious 2PC Approaches (constant round approaches)

- *Not so easy!*
- A spectrum of approaches!

	Interactive	Non-interactive
Not Practical	Yao+ZKP	Yao+ZKP
Asymptotically Efficient	*MPC-in-the-head [IKO+11]*	*MPC-in-the-head [IKO+11]*
Highly Practical		

Malicious 2PC Approaches (constant round approaches)

- *Not so easy!*
- A spectrum of approaches!

	Interactive	Non-interactive
Not Practical	Yao+ZKP	Yao+ZKP
Asymptotically Efficient	MPC-in-the-head [IKO+11]	MPC-in-the-head [IKO+11]
Highly Practical	*Cut-and-Choose [Lin13]*	

Malicious 2PC Approaches (constant round approaches)

- *Not so easy!*
- A spectrum of approaches!
- Can we make *Cut-and-Choose 2PC non-interactive*?

	Interactive	Non-interactive
Not Practical	Yao+ZKP	Yao+ZKP
Asymptotically Efficient	MPC-in-the-head [IKO+11]	MPC-in-the-head [IKO+11]
Highly Practical	Cut-and-Choose [Lin13]	Our Approach!

Outline

Contributions

- Practical NISC with *small overhead* based on *Cut-and-Choose 2PC*
  - As efficient as the state of the art [Lin13].
- *First* implementation of NISC.

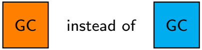
Making Cut-and-Choose Practical

- Cut-and-Choose Components
    - Checking circuits via Cut-and-Choose.
    - Input consistency.
    - Cheating Recovery.

Cut-and-Choose

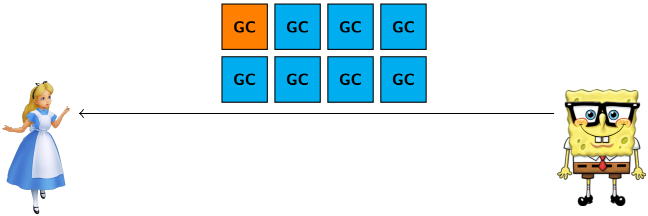
Cut-and-Choose Construction

- Attack?
  - *Malicious Sender* garbles *different circuits*.



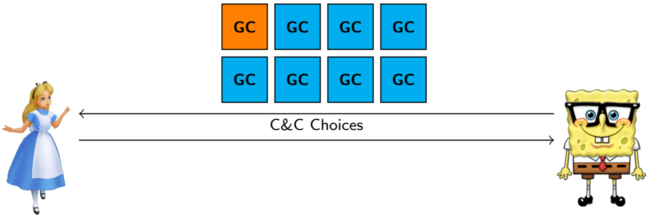
Cut-and-Choose Construction

- Attack?
  - Malicious Sender garbles different circuits.
- Cut-and-Choose Solution?
  - *Sender* garbles *many circuits* (i.e. around $\Omega(t)$ for 2^{-t} security).



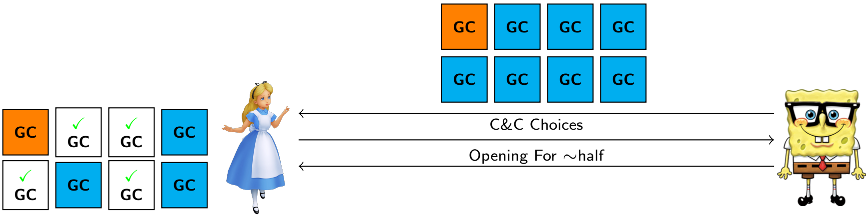
Cut-and-Choose Construction

- Attack?
  - Malicious Sender garbles different circuits.
- Cut-and-Choose Solution?
  - Sender garbles many circuits (i.e. around $\Omega(t)$ for 2^{-t} security).
  - *Receiver randomly chooses* half of them.



Cut-and-Choose Construction

- Attack?
  - Malicious Sender garbles different circuits.
- Cut-and-Choose Solution?
  - Sender garbles many circuits (i.e. around $\Omega(t)$ for 2^{-t} security).
  - Receiver randomly chooses half of them.
  - *Sender opens* the selected circuits.



Cut-and-Choose Construction

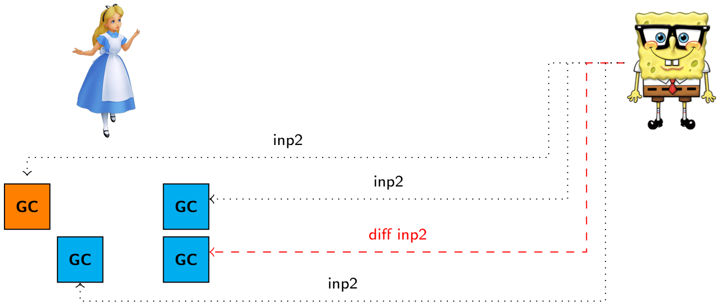
- Attack?
  - Malicious Sender garbles different circuits.
- Cut-and-Choose Solution?
  - Sender garbles many circuits (i.e. around $\Omega(t)$ for 2^{-t} security).
  - Receiver randomly chooses half of them.
  - Sender opens the selected circuits.
  - *Receiver evaluates* the other half and returns the majority result.

Cut-and-Choose Overhead

- Cost?
  - Adds *an extra round* (2 messages).
  - At least, an overhead of $3t$ in computation and communication.

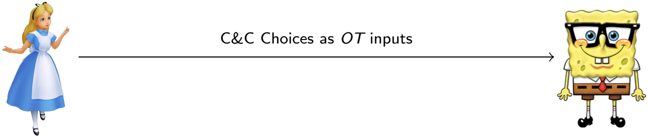
Cut-and-Choose Overhead

- Cost?
  - Adds an extra round (2 messages).
  - At least, an overhead of $3t$ in computation and communication.
- Problem: Sender's Input Consistency
  - Malicious Sender may send *different inputs* to *different circuits*.



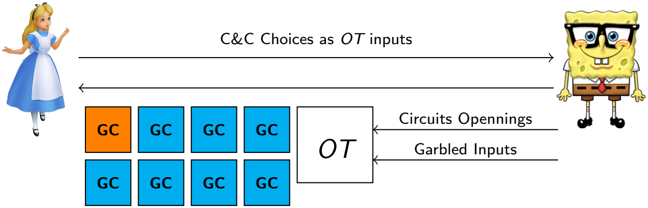
Non-interactive Cut-and-Choose

- Receiver's Message: Same as before



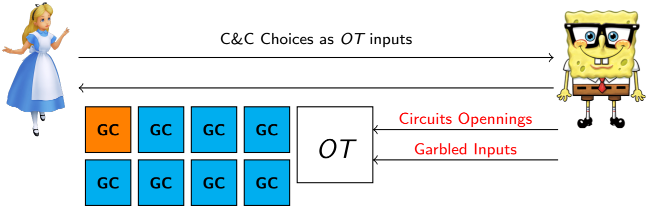
Non-interactive Cut-and-Choose

- Receiver's Message: Same as before
- Sender's Response:
  - Garbled circuits.
  - Circuit openings and garbled inputs as OT response.



Non-interactive Cut-and-Choose

- Receiver's Message: Same as before
- Sender's Response:
  - Garbled circuits.
  - Circuit openings and garbled inputs as OT response.
- Receiver's Computation: *For each circuit*
  - Receives *either* the opening
  - *or* the keys for evaluation



Input Consistency

Input Consistency

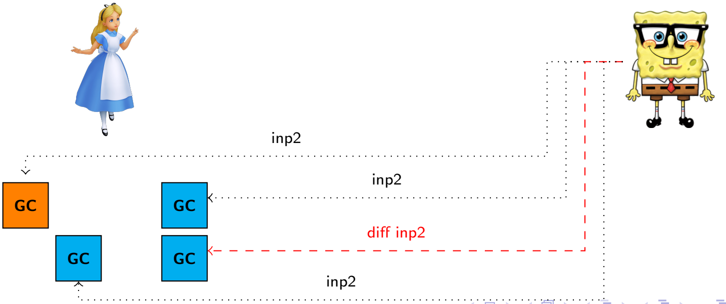
- *Receiver's* input consistency:
  - *Attack*: Selective OT Attack.

Input Consistency

- Receiver's input consistency:
  - Attack: Selective OT Attack.
  - *Defense*: Modified ideas of [LP11] and [SS11].

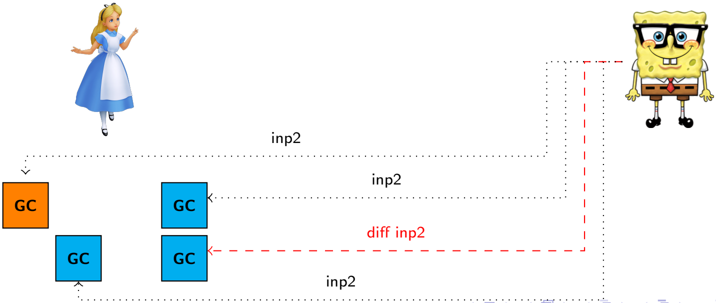
Input Consistency

- Receiver's input consistency:
  - Attack: Selective OT Attack.
  - Defense: Modified ideas of [LP11] and [SS11].
- *Sender's* input consistency:
  - *Attack*: Sender sends different inputs to different circuits.



Input Consistency

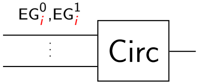
- Receiver's input consistency:
  - Attack: Selective OT Attack.
  - Defense: Modified ideas of [LP11] and [SS11].
- *Sender's* input consistency:
  - Attack: Sender sends different inputs to different circuits.
  - *Interactive Defense*: Different solutions with different costs!
  - *State of the art*: [MR13] and [sS13]



NISC Input Consistency

Sender

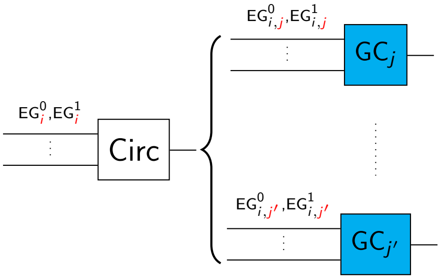
- Commits to original values.



NISC Input Consistency

Sender

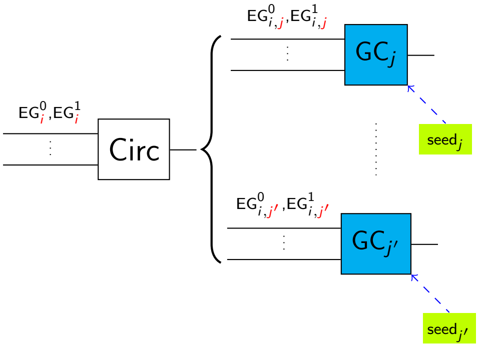
- Commits to original values.
- Garbles $3t$ copies of the circuit.



NISC Input Consistency

Sender

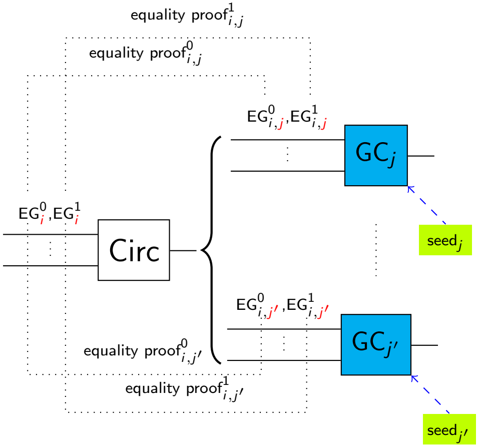
- Commits to original values.
- Garbles $3t$ copies of the circuit.
  - From different seeds.



NISC Input Consistency

Sender

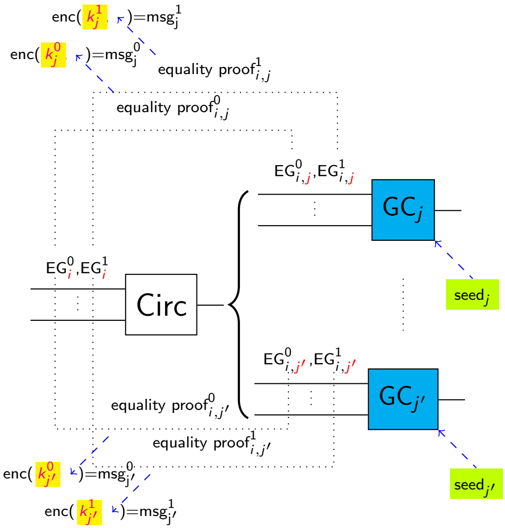
- Commits to original values.
- Garbles $3t$ copies of the circuit.
  - From different seeds.
- Generates equality proofs.



NISC Input Consistency

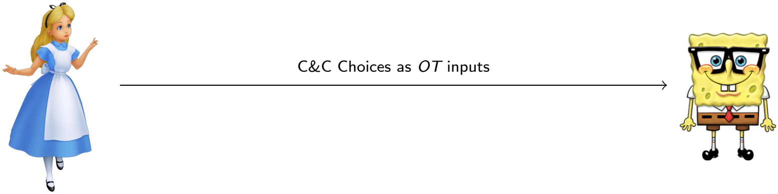
Sender

- Commits to original values.
- Garbles $3t$ copies of the circuit.
  - From different seeds.
- Generates equality proofs.
- Hides equality proofs.



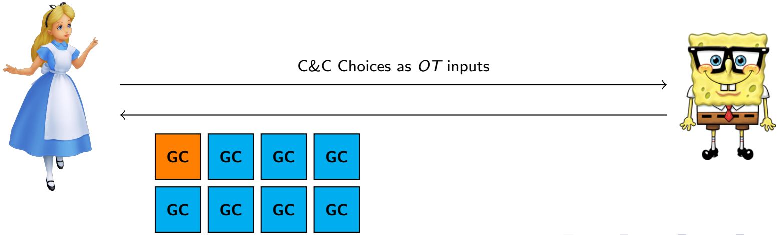
NISC Input Consistency

- First Message: The same as Cut-and-Choose.



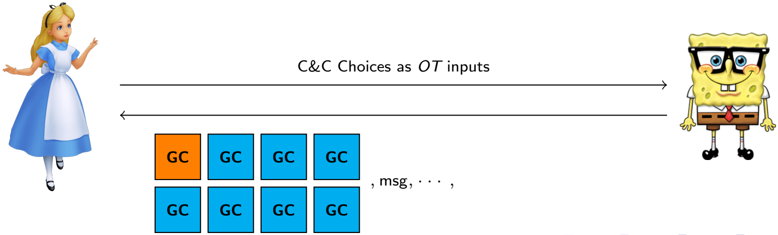
NISC Input Consistency

- First Message: The same as Cut-and-Choose.
- Second Message:
  - Garbled circuits



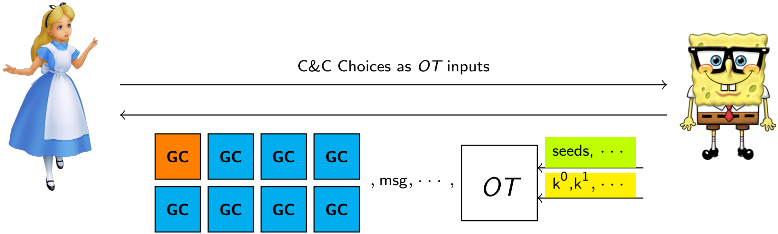
NISC Input Consistency

- First Message: The same as Cut-and-Choose.
- Second Message:
  - Garbled circuits
  - Encrypted proofs of equality.



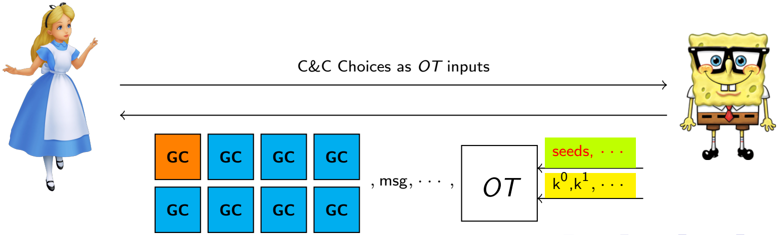
NISC Input Consistency

- First Message: The same as Cut-and-Choose.
- Second Message:
  - Garbled circuits
  - Encrypted proofs of equality.
  - OT response.



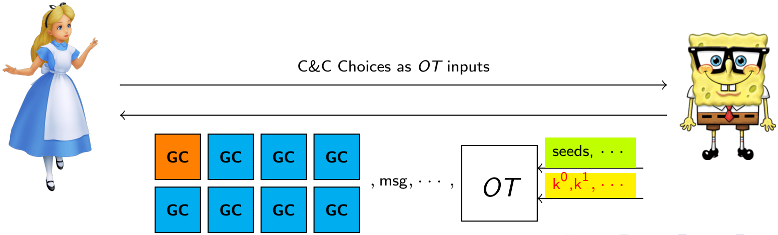
NISC Input Consistency

- First Message: The same as Cut-and-Choose.
- Second Message:
  - Garbled circuits
  - Encrypted proofs of equality.
  - OT response.
- Computation:
  - Opened circuits: Checks commitments.



NISC Input Consistency

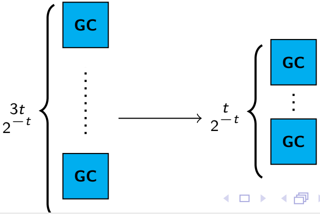
- First Message: The same as Cut-and-Choose.
- Second Message:
  - Garbled circuits
  - Encrypted proofs of equality.
  - OT response.
- Computation:
  - Opened circuits: Checks commitments.
  - Evaluated circuits: Checks equality.



Cheating Recovery

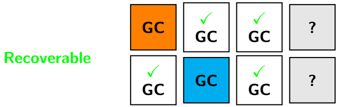
Cheating Recovery

- Goal
  - Reducing the $3t$ circuit overhead to t for security of 2^{-t}.
  - *No majority* checking.
  - At least *one correct* circuit.



Cheating Recovery

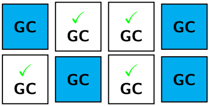
- The idea: Recover Sender's input
  - *two different* output wire values.



Cheating Recovery

- With t circuits, input *cannot* be recovered if
  - all evaluating are *the same* and *correct*.
    - Won't punish honest Sender.

Not Recoverable
Desirable!



Cheating Recovery

- With t circuits, input *cannot* be recovered if
- all evaluating are *the same* and *incorrect*.
    - 2^{-t} failure probability.

Not Recoverable
Not Desirable!

Cheating Recovery (cont.)

- How?
  - Use extra malicious 2PC [Lin13],
  - on a small circuit.

Cheating Recovery (cont.)

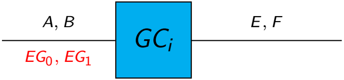
- How?
  - Use extra malicious 2PC [Lin13],
  - on a small circuit.
- cost?
  - *Extra rounds* for the extra 2PC.
  - Small computation and communication overhead for the extra 2PC.

NISC cheating recovery

- First message: Nothing!

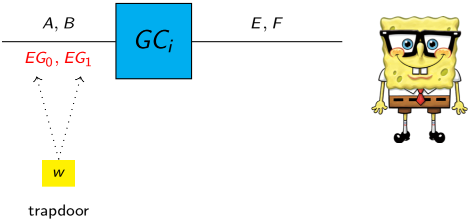
NISC cheating recovery

- First message: Nothing!
- Second message:
    - *Change* input labels to ElGamal commitments.



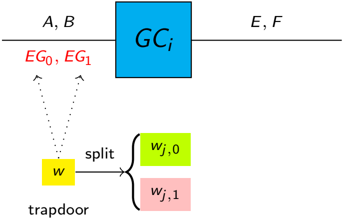
NISC cheating recovery

- First message: Nothing!
- Second message:
    - Change input labels to ElGamal commitments.
    - Assume a *trapdoor* w.



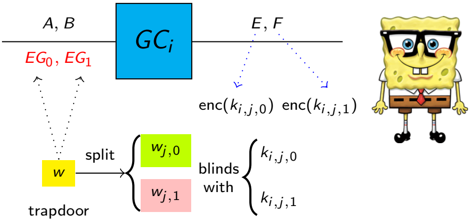
NISC cheating recovery

- First message: Nothing!
- Second message:
  - Change input labels to ElGamal commitments.
  - Assume a trapdoor w.
  - *Recover* trapdoor *(partly)*, using output labels.



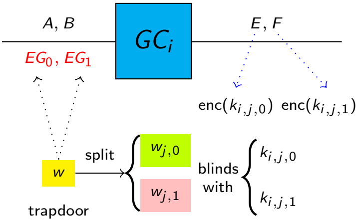
NISC cheating recovery

- First message: Nothing!
- Second message:
  - Change input labels to ElGamal commitments.
  - Assume a trapdoor w.
  - *Recover* trapdoor *(partly)*, using output labels.



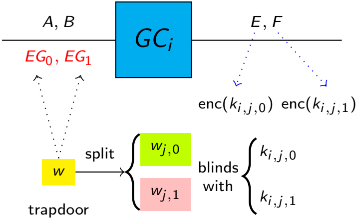
NISC cheating recovery

- First message: Nothing!
- Second message:
  - Change input labels to ElGamal commitments.
  - Assume a trapdoor w.
  - Recover trapdoor (partly), using output labels.
- Computation:
  - No cheating $\rightarrow$ learn *one* $w_{j,b}$.



NISC cheating recovery

- First message: Nothing!
- Second message:
  - Change input labels to ElGamal commitments.
  - Assume a trapdoor w.
  - Recover trapdoor (partly), using output labels.
- Computation:
  - No cheating → learn *one* $w_{j,b}$.
  - Cheating → learn *both* $w_{j,b}$.



Implementation

Results

- Experimented on a Linux VM on a laptop
  - 64bit, i7-4650U, CPU @ 1.70GHz and 5.4GB of RAM.
  - Used only one core.
  - Enabled AES-NI instructions set.
- AES circuit (with 8,492 non-XOR and 25,124 XOR gates).
  - $\sim$ 6.4 seconds
- Libraries used:
  - JustGarble [BHKR13], OpenSSL, RELIC-toolkit.

Observations

Module or part name	**Time**(sec.)	**Time**(sec.)
	AES circuit	*SHA256* circuit
First message	0.03	0.03
Second message	3.55	7.59
Receiver's computation	2.79	5.10
Cheating recovery	< 0.01	< 0.01
Total time	6.39	12.74
I/O time	0.53	4.89

Figure : For $t = 40$

- *Very small*
  - Garbling time.
  - Cheating recovery time.
- *Bottlenecks*
  - Exponentiations (Elliptic curve multiplication)
  - IO (used file for communications)

Observations

Module or part name	Time(sec.)	Time(sec.)
	AES circuit	*SHA256* circuit
Total time	6.39	12.74
I/O time	0.53	4.89
Non-I/O time	5.86	7.85

Figure : For $t = 40$

- *SHA256* garbled circuit size is $\sim$ *10 times larger*.
- Same input size, different output size
- Resulting in
  - Large increase in IO.
  - Small increase in non-IO.

Thank You!

References

Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway.
Efficient garbling from a fixed-key blockcipher.
In *IEEE S&P*, 2013.

Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai.
Efficient non-interactive secure computation.
In *EUROCRYPT*, pages 406–425. Springer-Verlag, 2011.

Yehuda Lindell.
Fast cut-and-choose based protocols for malicious and covert adversaries.
In *CRYPTO*, pages 1–17, 2013.

Yehuda Lindell and Benny Pinkas.
Secure two-party computation via cut-and-choose oblivious transfer.
In *TCC*, pages 329–346. Springer, 2011.

Payman Mohassel and Ben Riva.
Garbled circuits checking garbled circuits: More efficient and secure two-party computation.
In *CRYPTO*, pages 36–53, 2013.

Chris Peikert, Vinod Vaikuntanathan, and Brent Waters.
A framework for efficient and composable oblivious transfer.
In *CRYPTO*, 2008.

Abhi Shelat and Chih-Hao Shen.
Two-output secure computation with malicious adversaries.
In *EUROCRYPT*, pages 386–405. Springer, 2011.

abhi shelat and Chih-hao Shen.
Fast two-party secure computation with minimal assumptions.
In *CCS*, pages 523–534. ACM, 2013.

resources...

- SHA256 circuit (with 194,083 non-XOR gates and 42,029 XOR gates).
- AES circuit (with 8,492 non-XOR and 25,124 XOR gates).