TU Graz

SCIENCE
PASSION
TECHNOLOGY

Multiparty Computation - Advanced Topics

Daniel Kales

Applied Cryptography 2 – ST 2020

Outline

Garbled Circuit Optimizations

Security against Malicious Adversaries

Use Cases for MPC Protocols

- Private Set Intersection
- Zero-Knowledge Proofs

Garbled Circuit Optimizations

Garbled Circuits

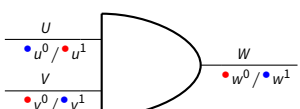
- Remember Yao's Garbled Circuits
  - Alice garbles a circuit $\mathcal{C}$ and sends it to Bob
  - Alice sends the wire labels corresponding to her input
  - Bob receives wire labels for his input using oblivious transfer
  - Bob evaluates the circuit
  - Parties communicate to learn value of output labels

Performance of Garbled Circuits

Method	Size/Gate	Enc. (Garbler)	Enc. (Receiver)
Classic	4	4	4

Point-and-Permute (P&P) [1]

- Idea: Why perform unnecessary decryptions at the evaluator?
  - Want to tell evaluator which element of table to decrypt
  - …without leaking information!
- P&P: Random coloring of wire lables



Garbled Table

●●	$\mathbb{E}_{u^1,v^0}(\bullet w^0)$
●●	$\mathbb{E}_{u^1,v^1}(\bullet w^1)$
●●	$\mathbb{E}_{u^0,v^1}(\bullet w^0)$
●●	$\mathbb{E}_{u^0,v^0}(\bullet w^0)$

Point-and-Permute (cont.)

Point-and-Permute for Garbled Circuits

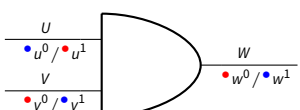
- Assign each pair of wire labels colors at random
- Store color in MSB of wire label
- Permute garbled table in canonical order
- Intuitivly, because colors are chosen at random, whole table is randomly permuted
- Evaluator knows exactly which entry to decode

Performance of Garbled Circuits

Method	Size/Gate	Enc. (Garbler)	Enc. (Receiver)
Classic	4	4	4
P&P	4	4	1

Garbled Row Reduction (GRR3) [5]

- Until now, all wire labels are chosen at random
  - Can we relax this constraint?
- Idea: Choose output label so that encryption is 0



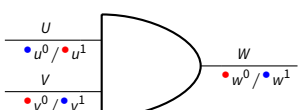
Garbled Table

●●	$\mathbb{E}_{u^1,v^0}(\bullet w^0)$
●●	$\mathbb{E}_{u^1,v^1}(\bullet w^1)$
●●	$\mathbb{E}_{u^0,v^1}(\bullet w^0)$
●●	$\mathbb{E}_{u^0,v^0}(\bullet w^0)$

$$w^0 \stackrel{!}{=} \mathbb{E}^{-1}_{u^1,v^0}(0) \Rightarrow \mathbb{E}_{u^1,v^0}(w^0) = 0$$

Garbled Row Reduction (GRR3) [5]

- Until now, all wire labels are chosen at random
  - Can we relax this constraint?
- Idea: Choose output label so that encryption is 0



Garbled Table

●●	0
●●	$\mathbb{E}_{u^1,v^1}(\bullet w^1)$
●●	$\mathbb{E}_{u^0,v^1}(\bullet w^0)$
●●	$\mathbb{E}_{u^0,v^0}(\bullet w^0)$

We don't need to send 0!

Performance of Garbled Circuits

Method	Size/Gate	Enc. (Garbler)	Enc. (Receiver)
Classic	4	4	4
P&P	4	4	1
GRR3	3	4	1

Free-XOR

- Biggest optimization in history of garbled circuits
- Idea: Choose wire labels according to rule, not at random

Free-XOR [4]

- Choose global difference Δ
- Invariant for all wires w_i:
  - $w_i^1 = w_i^0 \oplus \Delta$
- For all XOR Gates ($w = u \oplus v$):
  - Set $w^0 = u^0 \oplus v^0$

Free-XOR (cont.)

u^0	v^0	$w^0 = u^0 \oplus v^0$
$u^1 = u^0 \oplus \Delta$	$v^1 = v^0 \oplus \Delta$	$w^1 = w^0 \oplus \Delta$

$\oplus$	$v = 0$	$v = 1$
$u = 0$	$u^0 \oplus v^0 = w^0$	$u^0 \oplus v^1 = u^0 \oplus (v^0 \oplus \Delta) = w^0 \oplus \Delta = w^1$
$u = 1$	$u^1 \oplus v^0 = (u^0 \oplus \Delta) \oplus v^0 = w^0 \oplus \Delta = w^1$	$u^1 \oplus v^1 = (u^0 \oplus \Delta) \oplus (v^0 \oplus \Delta) = u^0 \oplus v^0 = w^0$

Free-XOR (cont.)

- XOR gates are now free to evaluate
- Evaluator just XOR's the wire labels
  - No tranmission of garbled table
  - No encryptions for both parties
- New optimization goal:
  - Reduce number of AND gates
  - Even at cost of much more XOR gates
  - …at least where possible
- Compatible with GRR3!

Performance of Garbled Circuits

Method	Size/Gate	Enc. (Garbler)	Enc. (Receiver)
Classic	4	4	4
P&P	4	4	1
GRR3	3	4	1
Free-XOR	3 (AND) 0 (XOR)	4 (AND) 0 (XOR)	1 (AND) 0 (XOR)

Performance of Garbled Circuits

Method	Size/Gate	Enc. (Garbler)	Enc. (Receiver)
Classic	4	4	4
P&P	4	4	1
GRR3	3	4	1
Free-XOR	3 (AND) 0 (XOR)	4 (AND) 0 (XOR)	1 (AND) 0 (XOR)
GRR2	2 (AND) 2 (XOR)	4 (AND) 4 (XOR)	1 (AND) 1 (XOR)
Halves-Gate [7]	2 (AND) 0 (XOR)	4 (AND) 0 (XOR)	2 (AND) 0 (XOR)

Security against Malicious Adversaries

Malicious Parties

Attack Vector 1

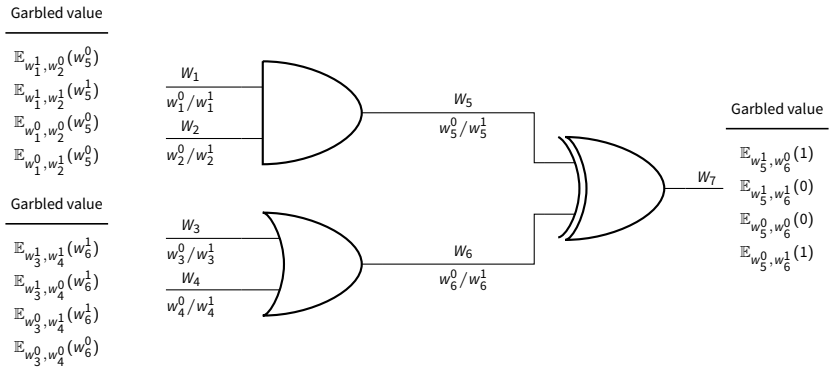
- Alice could send arbitrary circuit $\mathcal{C}$
  - Why not just build circuit that evaluates $F(x, y) = y$?
  - Bob does not know all labels, he cannot verify the circuit
  - Alice learns Bob's input y

Attack Vector 2

- Alice could send wrong/invalid input labels to Bob
  - Fault attack: Send 1 valid and 1 invalid wire label
  - Based on if Bob aborts, Alice can learn his input
  - Bob cannot verify beforehand if label is valid

Cut-and-Choose

- Remember: Garbled circuits

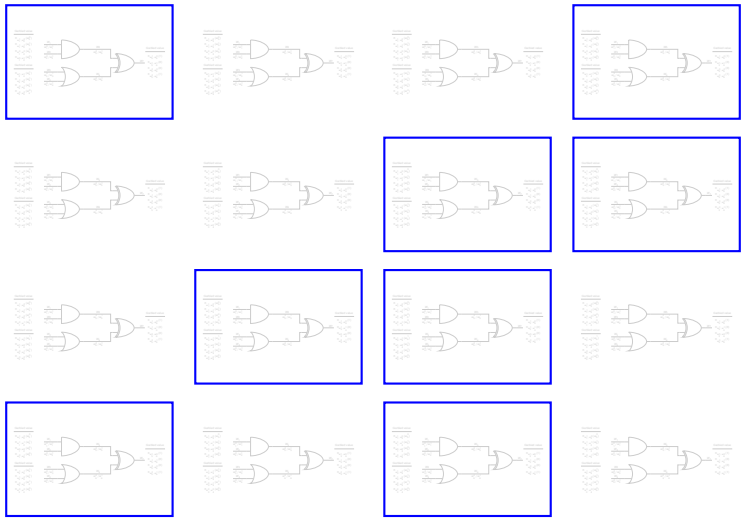


Cut-and-Choose

- Alice creates s garbled circuits and sends them to Bob

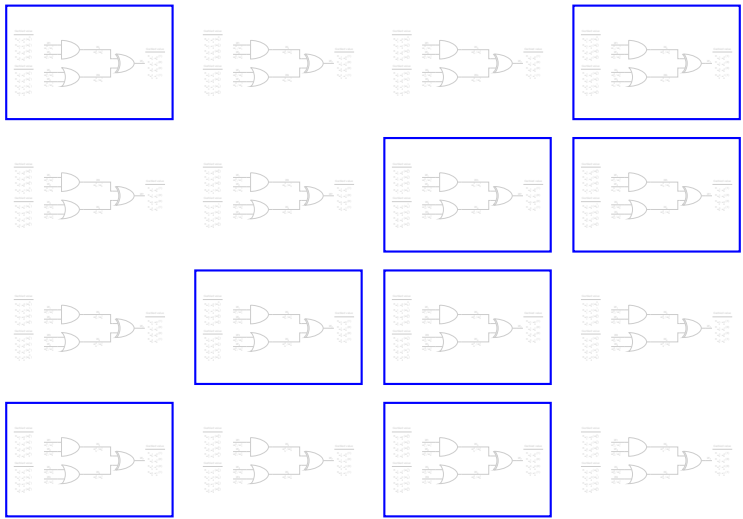
Cut-and-Choose

- Bob requests Alice to open randomly chosen circuits



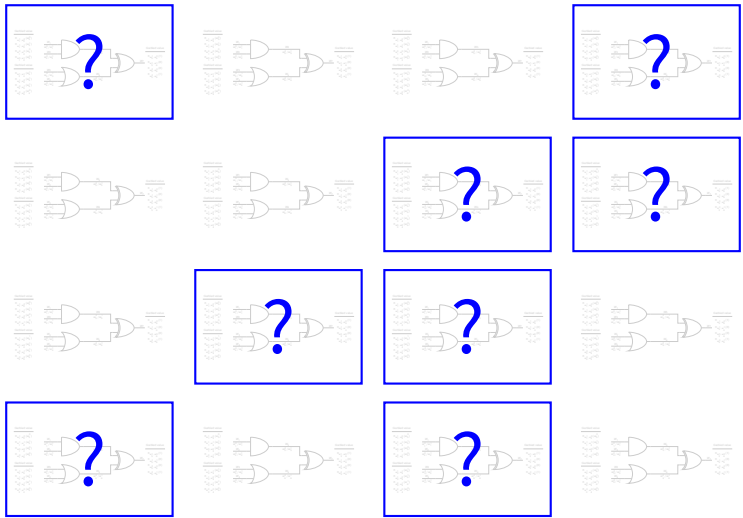
Cut-and-Choose

- Alice opens the chosen circuits by sending all wire labels



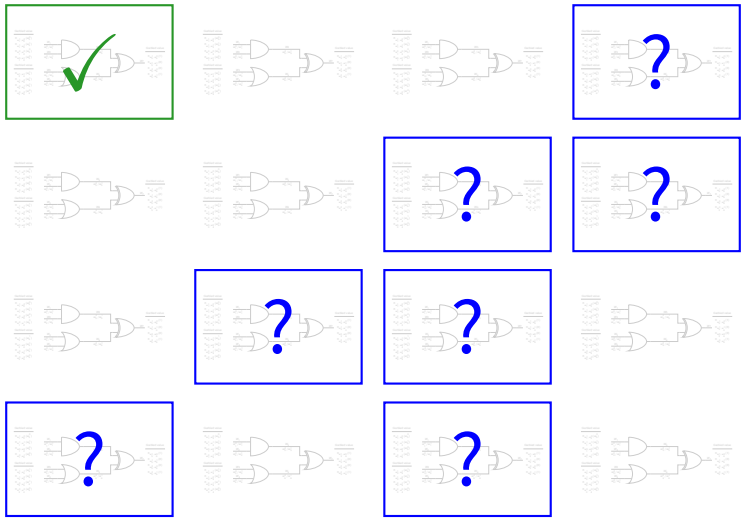
Cut-and-Choose

- Bob verifies the chosen circuits for validity



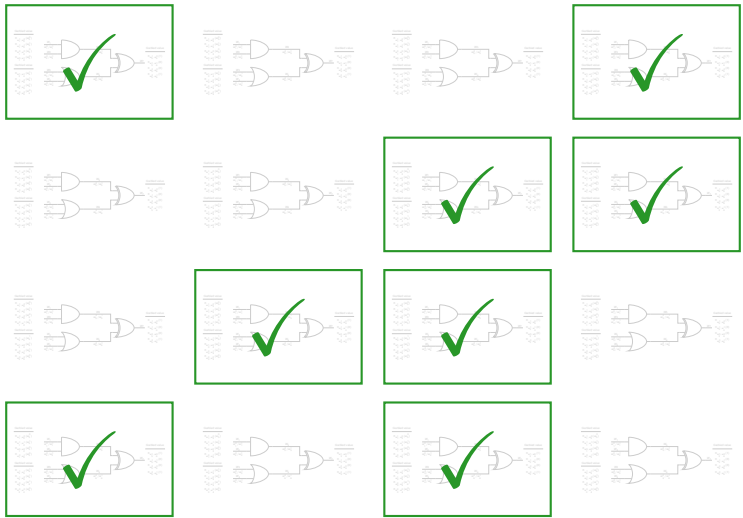
Cut-and-Choose

- Bob verifies the chosen circuits for validity



Cut-and-Choose

- Bob verifies the chosen circuits for validity

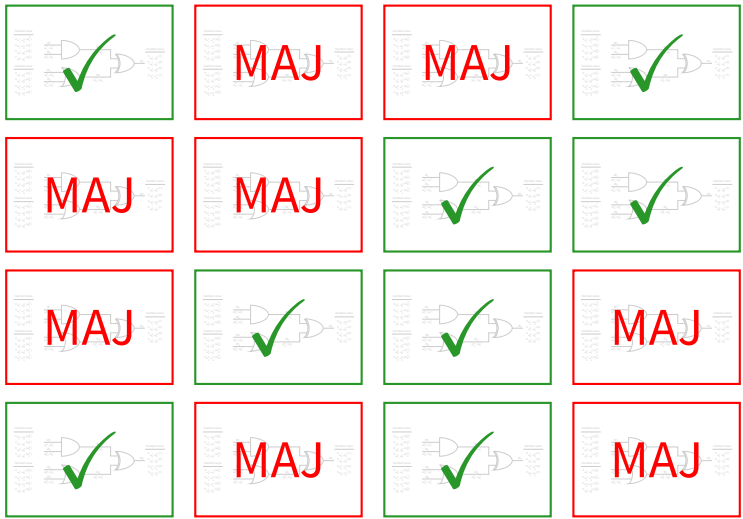


Cut-and-Choose

- Bob evaluates the rest of the circuits

Cut-and-Choose

- Bob chooses the output based on the majority (Why?)



Cut-and-Choose (cont.)

Cut-and-Choose for Garbled Circuits

1. Alice creates s garbled circuits
2. Bob requests Alice to open randomly chosen circuits ($\approx 60\%$)
3. Alice opens the chosen circuits by sending all wire labels
4. Bob verifies the chosen circuits for validity
5. Bob evaluates the rest of the circuits
6. Bob outputs the majority result
   - Output of $\perp$ can leak information!

Cut-and-Choose (cont.)

- Can Alice still cheat (send a wrong circuit)?
  - She needs to guess which circuits will be opened
  - All opened circuits would need to be valid
  - The majority of the evaluated circuits need to be malicious
  - Probability for guess that allows cheating: $2^{-0.32s}$
- Alice can still cheat in the OT phase
  - Send valid wire label for choice 0
  - Send invalid wire label for choice 1
  - If Bob aborts, Alice knows his input was 1 ($\rightarrow$ Majority!)
- Maliciously Secure OT + Cut-and-Choose for inputs (out of scope)

Cut-and-Choose (cont.)

- Can Alice still cheat (send a wrong circuit)?
  - She needs to guess which circuits will be opened
  - All opened circuits would need to be valid
  - The majority of the evaluated circuits need to be malicious
  - Probability for guess that allows cheating: $2^{-0.32s}$
- Alice can still cheat in the OT phase
  - Send valid wire label for choice 0
  - Send invalid wire label for choice 1
  - If Bob aborts, Alice knows his input was 1 ($\rightarrow$ Majority!)
- Maliciously Secure OT + Cut-and-Choose for inputs (out of scope)

Cut-and-Choose (cont.)

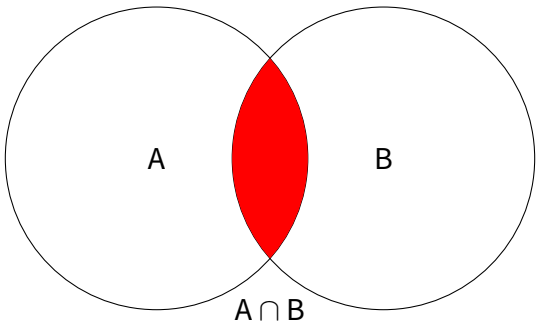
- Can Alice still cheat (send a wrong circuit)?
  - She needs to guess which circuits will be opened
  - All opened circuits would need to be valid
  - The majority of the evaluated circuits need to be malicious
  - Probability for guess that allows cheating: $2^{-0.32s}$
- Alice can still cheat in the OT phase
  - Send valid wire label for choice 0
  - Send invalid wire label for choice 1
  - If Bob aborts, Alice knows his input was 1 ($\rightarrow$ Majority!)
- Maliciously Secure OT + Cut-and-Choose for inputs (out of scope)

Use Cases for MPC Protocols

Private Set Intersection

Private Set Intersection

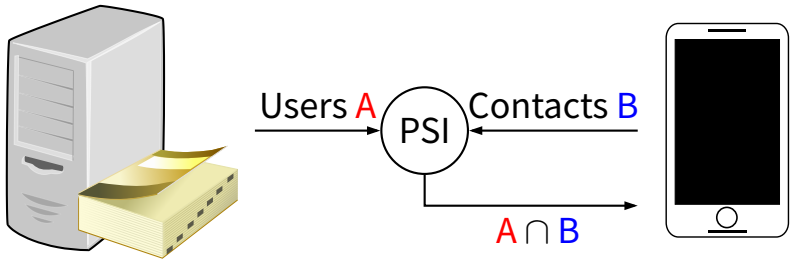
- Compute intersection of two sets
- Privacy-preserving



Private Contact Discovery

- Imagine a messaging app:
  - New user joins service
  - Wants to know if any of his contacts are using the service
  - Server does not want to send him a list of all users
  - Client does not want to send him a list of all contacts
- Goal: Privately compute intersection

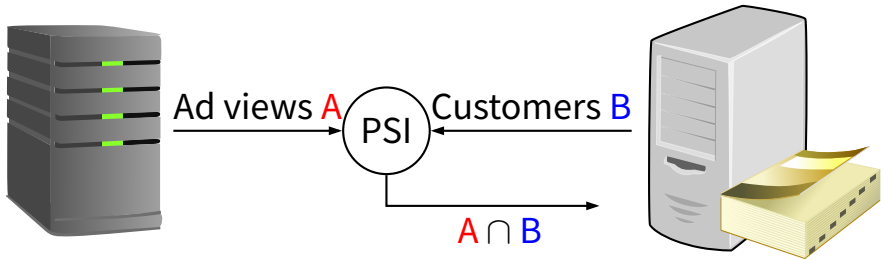
Private Contact Discovery (cont.)



Advertising Efficiency

- Advertising Companies
  - Keep record of who watches advertisements
  - Companies want to know how many of their customers watched ad
  - Again, both parties do not want to disclose all their data
  - Legal reasons

Advertising Efficiency (cont.)



A First Attempt

- Naive Solution
  - Each party hashes their inputs
  - Exchange hash sets and perform intersection
- Problems?
  - Very easy to brute-force offline
  - Especially with low-entropy inputs

A First Attempt

- Naive Solution
  - Each party hashes their inputs
  - Exchange hash sets and perform intersection
- Problems?
  - Very easy to brute-force offline
  - Especially with low-entropy inputs

Circuit-Based PSI

- We know how to securely evaluate a circuit
  - Last lecture
- Can we use this to build a protocol for PSI?
- Yes!
  - Sort-Compare-Shuffle
  - Oblivious Pseudorandom Function

Circuit-Based PSI

- We know how to securely evaluate a circuit
  - Last lecture
- Can we use this to build a protocol for PSI?
- Yes!
  - Sort-Compare-Shuffle
  - Oblivious Pseudorandom Function

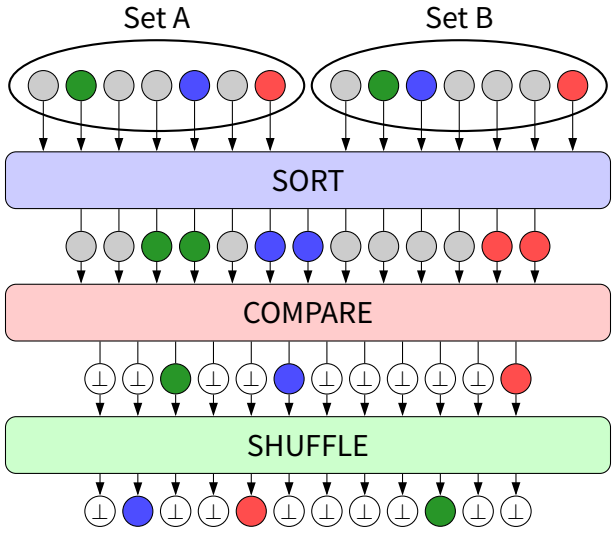
Sort-Compare-Shuffle

- Basic idea:
  - Parties do not have duplicate items in their own sets
  - If element appears twice in $A \cup B$, it must be in $A \cap B$

PSI using a Sort-Compare-Shuffle circuit [2]

- Alice and Bob evaluate a circuit that:
  - Sorts combined input set
  - Checks if adjacent elements are equal
  - If yes, puts element in output set
  - Shuffles output set to remove leakage based on order

Sort-Compare-Shuffle (cont.)



Sort-Compare-Shuffle (cont.)

- Size of different circuits (σ-bit elements):
  - Sort: $\mathcal{O}(\sigma n \log(n))$
  - Compare: $\mathcal{O}(\sigma n)$
  - Shuffle: $\mathcal{O}(\sigma n \log(n))$
- Total: $\mathcal{O}(\sigma n \log(n))$

Oblivious Pseudorandom Function (OPRF)

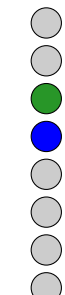
- Idea: Use encryption of elements for comparison
  - Cannot be brute-forced offline, only one party knows key

PSI using OPRF Evaluation [6]

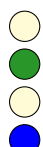
- Alice samples a key k and encrypts each item in her set
- Alice sends her encrypted set to Bob
- Alice and Bob evaluate the encryption circuit
  - Alice inputs the key k
  - Bob inputs his element y_i
  - Only Bob learns $E(k, y_i)$
- Bob performs the intersection of the two encrypted sets

OPRF Evaluation (cont.)

- Let’s look at an example evaluation



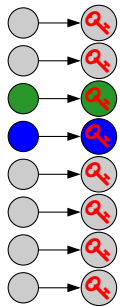
Alice



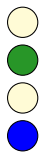
Bob

OPRF Evaluation (cont.)

- Alice encrypts her items with a randomly chosen key



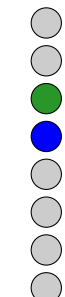
Alice



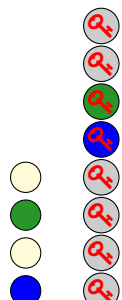
Bob

OPRF Evaluation (cont.)

- Alice sends her encrypted set to Bob



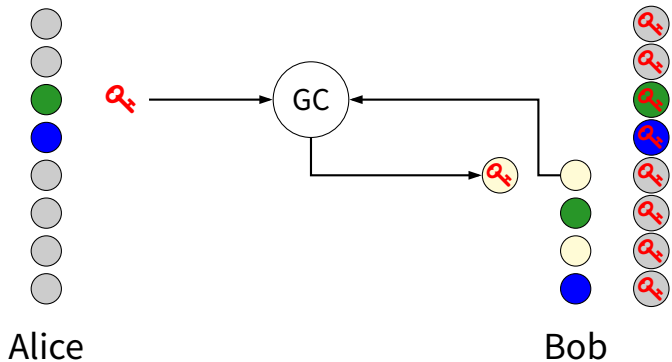
Alice



Bob

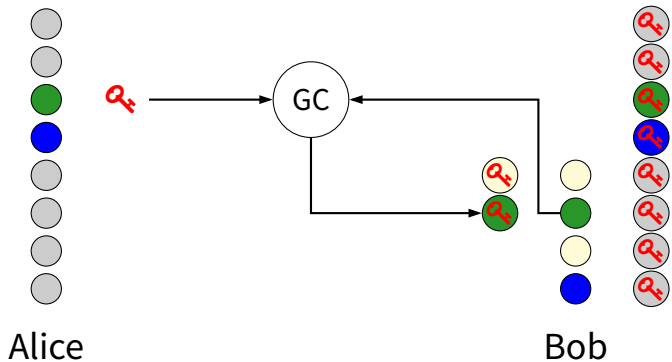
OPRF Evaluation (cont.)

- Bob and Alice evaluate the circuit to encrypt Bob's items



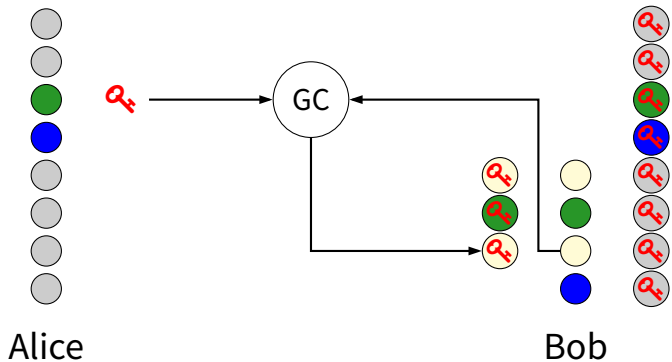
OPRF Evaluation (cont.)

- Bob and Alice evaluate the circuit to encrypt Bob's items



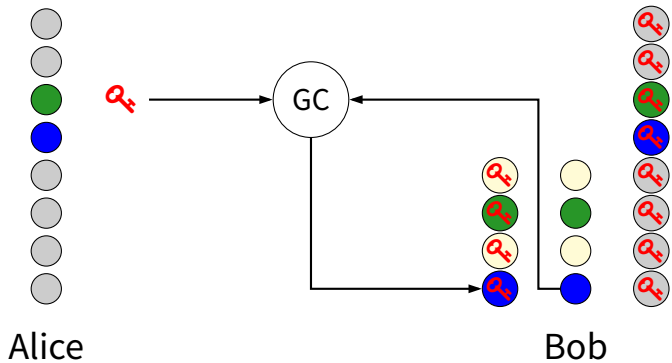
OPRF Evaluation (cont.)

- Bob and Alice evaluate the circuit to encrypt Bob's items



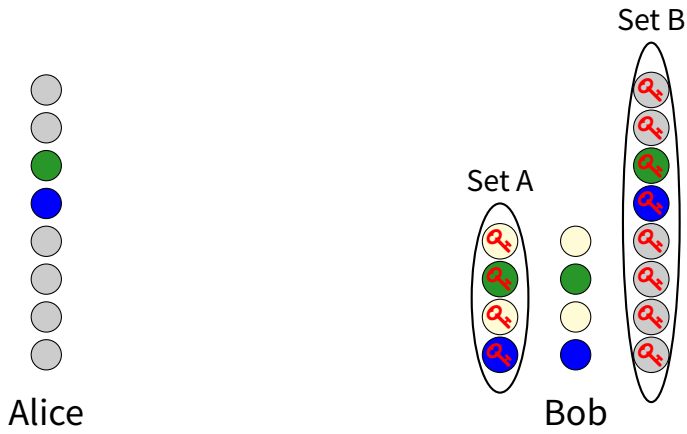
OPRF Evaluation (cont.)

- Bob and Alice evaluate the circuit to encrypt Bob's items



OPRF Evaluation (cont.)

- Bob calculates the intersection of the encrypted sets



OPRF Evaluation (cont.)

- Bob knows the items corresponding to the result

OPRF Evaluation (cont.)

- Especially suited for unequal set sizes!
- Assume $|S_A| \gg |S_B|$:
  - Alice's larger set can be encrypted without MPC
  - Only Bob's smaller set has to be evalated using MPC
- Example: Private Contact Discovery
  - Server database much larger than user contact list

OPRF Evaluation (cont.)

- Circuit size depends on encryption function!
- Specialized ciphers: LowMC, MiMC
  - Low number of AND gates
  - Smaller circuit size
  - Less communication, faster runtime
- Comparison of encryption circuits
  - AES: 5120 AND gates $\rightarrow$ $\approx$ 170 KB
  - LowMC: 624 AND gates $\rightarrow$ $\approx$ 20 KB

Zero-Knowledge Proofs

Zero-Knowledge Proofs

- Useful proof in many protocols:
  - “I know a preimage m of $H(m)$”
  - “I know a key k that encrypts a message m to $E(k, m)$”
- Structure of hash functions and block ciphers suited for garbled circuits
- Not well suited for algebraic relations
  - “I know a discrete logarithm x so that $y = g^x$”
  - Very inefficient circuit
  - Much better solutions exist

Zero-Knowledge Proofs

- Useful proof in many protocols:
  - “I know a preimage m of $H(m)$”
  - “I know a key k that encrypts a message m to $E(k, m)$”
- Structure of hash functions and block ciphers suited for garbled circuits
- Not well suited for algebraic relations
  - “I know a discrete logarithm x so that $y = g^x$”
  - Very inefficient circuit
  - Much better solutions exist

Zero-Knowledge Proofs: Required Properties

- Completeness:
  If Alice knows the secret, she can carry out the protocol successfully.
- Soundness:
  If Eve can convince Bob that she is Alice with non-negligible probability, then Eve can extract Alice's secret.
- Zero-knowledge:
  Even a dishonest Bob learns nothing except 1 bit (he is talking to Alice). He could have produced himself all the other information he obtains during the protocol (simulatable).

Observation: A successful exchange between A and B means nothing to an observer O. B cannot later prove to O that A knows something. O cannot tell whether A and B faked the exchange.

Zero-Knowledge Proofs: Required Properties

- Completeness:
  If Alice knows the secret, she can carry out the protocol successfully.
- Soundness:
  If Eve can convince Bob that she is Alice with non-negligible probability, then Eve can extract Alice's secret.
- Zero-knowledge:
  Even a dishonest Bob learns nothing except 1 bit (he is talking to Alice). He could have produced himself all the other information he obtains during the protocol (simulatable).

Observation: A successful exchange between A and B means nothing to an observer O. B cannot later prove to O that A knows something. O cannot tell whether A and B faked the exchange.

Zero-Knowledge Proofs (cont.)

"I know a preimage m of $H(m)$"

Zero-Knowledge Proofs using Garbled Circuits (v1)

1. Bob builds a garbled circuit that evaluates H
   - Bob has no input to circuit
2. Alice uses Oblivious Transfer to get input labels for m
3. Alice evaluates circuit and sends output labels to Bob
4. Bob verifies that output labels correspond to $H(m)$

Zero-Knowledge Proofs (cont.)

"I know a preimage m of $H(m)$"

Zero-Knowledge Proofs using Garbled Circuits (v2)

1. Bob builds a garbled circuit that evaluates H
2. Alice uses Oblivious Transfer to get input labels for m
3. Bob opens the circuit to show he did not cheat
   - Bob sends all wire labels
4. Alice verifies and evaluates the circuit and sends Bob output labels
5. Bob verifies that output labels correspond to $H(m)$

Zero-Knowledge Proofs (cont.)

"I know a preimage m of $H(m)$"

Zero-Knowledge Proofs using Garbled Circuits [3]

1. Bob builds a garbled circuit that evaluates H
2. Alice uses Oblivious Transfer to get input labels for m
3. Alice evaluates circuit and commits to output labels
4. Bob opens the circuit to show he did not cheat
5. If Alice is satisfied, she sends Bob output labels
6. Bob verifies that output labels correspond to $H(m)$ and commitments

Summary

Summary

- Garbled Circuit Optimizations
  - P&P, GRR3/2, Free-XOR, Halves-Gate
- Active Security for Garbled Circuits
  - Cut-and-Choose
- Use-Cases for MPC
  - Private Set Intersection
    - Sort-Compare-Shuffle
    - Oblivious Pseudorandom Function Evaluation
  - Zero-Knowledge Proofs

Questions you should be able to answer

1. Describe the Free-XOR optimization for Garbled Circuits.
2. What is the cut-and-choose approach for garbled circuits? Describe the steps involved in proving honest behaviour.
3. What is Private Set Intersection? Give a use case for PSI. How can one implement PSI using garbled circuits?
4. Describe how to build a zero-knowledge proof proving that one party knows a preimage m that hashes to a public value h, e.g., $h = \text{SHA-256}(m)$.

Bibliography I

[1] Donald Beaver, Silvio Micali, and Phillip Rogaway. **The Round Complexity of Secure Protocols (Extended Abstract)**. STOC. ACM, 1990, pp. 503–513.

[2] Yan Huang, David Evans, and Jonathan Katz. **Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?** NDSS. The Internet Society, 2012.

[3] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. **Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently**. ACM Conference on Computer and Communications Security. ACM, 2013, pp. 955–966.

[4] Vladimir Kolesnikov and Thomas Schneider. **Improved Garbled Circuit: Free XOR Gates and Applications**. ICALP (2). Vol. 5126. Lecture Notes in Computer Science. Springer, 2008, pp. 486–498.

[5] Moni Naor, Benny Pinkas, and Reuban Sumner. **Privacy preserving auctions and mechanism design**. EC. 1999, pp. 129–139.

Bibliography II

[6] Benny Pinkas et al. **Secure Two-Party Computation Is Practical.** ASIACRYPT. Vol. 5912. Lecture Notes in Computer Science. Springer, 2009, pp. 250–267.

[7] Samee Zahur, Mike Rosulek, and David Evans. **Two Halves Make a Whole - Reducing Data Transfer in Garbled Circuits Using Half Gates.** EUROCRYPT (2). Vol. 9057. Lecture Notes in Computer Science. Springer, 2015, pp. 220–250.