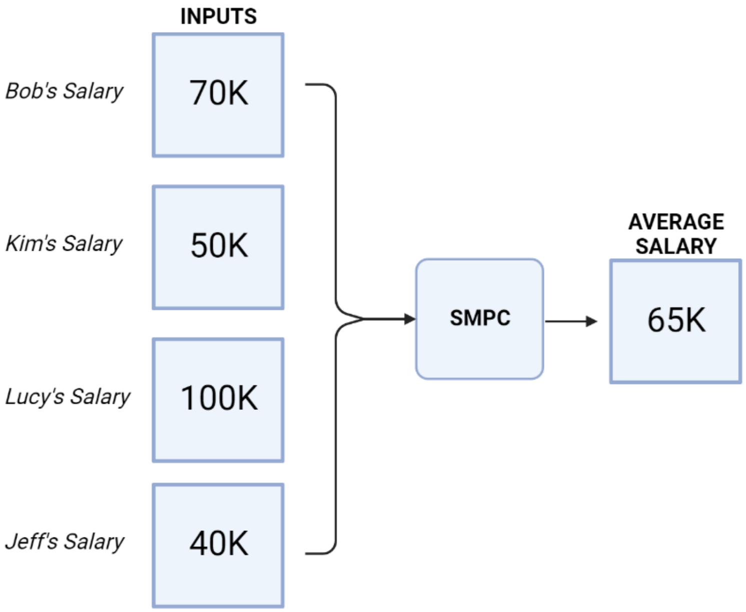
Making Secure Multi-Party Computation Scaleable

Nicholas Gilbertson
gilb057@morris.umn.edu
University of Minnesota Morris
October 27th

Introduction: The Salary Problem

The Salary Problem:

- We need to compute the average salary of people in a room
- We need each participant's salary to be private
- We can't use any shared trusted entity
- The solution: Secure Multi-Party Computation (SMPC)



Talk Outline

1. *INTRODUCTION*
    a. *What is SMPC?*
    b. *Implementations of SMPC*
    c. *GCE, nanoPI, and Conclave*
2. BACKGROUND
    a. Security Classifications of SMPC
3. SMPC PROTOCOLS
    a. Garbled Circuit Evaluation
4. SMPC SCALING
    a. Scaling Issues
    b. nanoPI
    c. Conclave
5. CONCLUSION

Introduction: What is SMPC?

Secure Multi-Party Computation (SMPC):

- Computation performed by multiple parties while keeping inputs private
- Arbitrary number of participants
- Arbitrary type of computation
- Privacy preservation dependent on threshold for given SMPC protocol
- Net based local decentralized process

Introduction: Implementations of SMPC

All participants see result:

- Auctions
- Privacy-preserving Machine Learning
- Poker without a trusted third party

Some participants see result:

- Financial data analysis
- Privacy-preservation in medical research

https://uxwing.com/auction-icon/

https://www.dreamstime.com/machine-learning-icon

https://stock.adobe.com/search?k=playing%20card%20symbols%20vector

Introduction: GCE, nanoPI, and Conclave

Garbled Circuit Evaluation (GCE):

- Popular implementation of SMPC
- Secure two-party computation
- Computes boolean circuits
- Configurable reception of results
- Different security level configurations

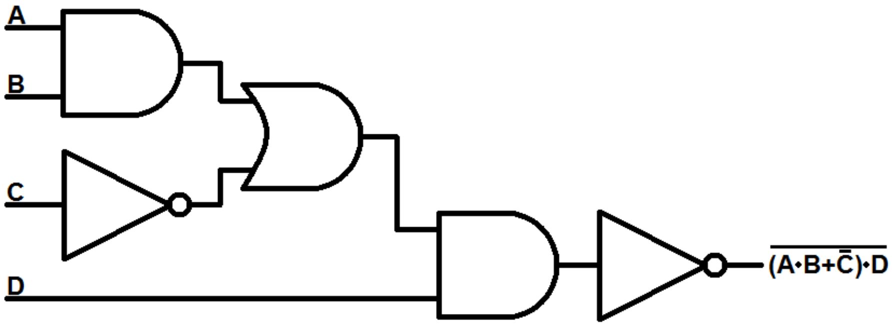


Figure: Example of a boolean circuit

Introduction: GCE, nanoPI, and Conclave

nanoPI:

- Based off state of the art SMPC protocols
- Attempts to fix scaling issues within SMPC
- Is a highly secure SMPC protocol

Introduction: GCE, nanoPI, and Conclave

Conclave:

- Query Compiler used to speed up existing SMPC implementations
- Allows for parallel processing of steps to improve efficiency
- Mid-level security

Talk Outline

1. INTRODUCTION
    a. What is SMPC?
    b. Implementations of SMPC
    c. GCE, nanoPI, and Conclave
2. *BACKGROUND*
    a. *Security Classifications of SMPC*
3. SMPC PROTOCOLS
    a. Garbled Circuit Evaluation
4. SMPC SCALING
    a. Scaling Issues
    b. nanoPI
    c. Conclave
5. CONCLUSION

Background: Security Classifications within SMPC

- Defends against “Semi-Honest Adversaries”
    - Passive Security
- Defends against “Malicious Adversaries”
    - Covert Security
    - Active Security

Talk Outline

1. INTRODUCTION
    a. What is SMPC?
    b. Implementations of SMPC
    c. GCE, nanoPI, and Conclave
2. BACKGROUND
    a. Security Classifications of SMPC
3. *SMPC PROTOCOLS*
    a. *Garbled Circuit Evaluation*
4. SMPC SCALING
    a. Scaling Issues
    b. nanoPI
    c. Conclave
5. CONCLUSION

SMPC Protocols: Garbled Circuit Evaluation

- Secure two-party computation used to construct Boolean Circuits

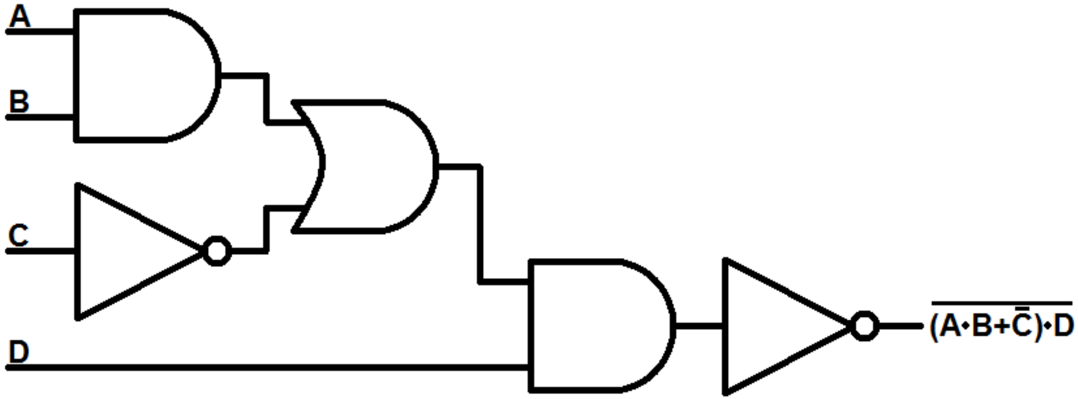


Figure: Example of a boolean circuit

SMPC Protocols: Garbled Circuit Evaluation

Garbled Circuit Evaluation Mechanics:

- Given participants Alice and Bob, their inputs $\{i_A, i_B\}$, and a function $f(\,)$
- GCE offers a way to compute $f(i_A, i_B)$ privately
- Given that Alice is the "Circuit Generator", Bob will be the "Circuit Evaluator"
- Alice creates a "Garbled Representation" of the circuit
- Bob is tasked with evaluating the inputs over this unreadable Garbled Circuit

SMPC Protocols: Garbled Circuit Evaluation

"1-out-of-2 Oblivious Transfer"

SMPC Protocols: Garbled Circuit Evaluation

"1-out-of-2 Oblivious Transfer"

- The Circuit Generator, Alice, has two strings, $\{s_0, s_1\}$

SMPC Protocols: Garbled Circuit Evaluation

"1-out-of-2 Oblivious Transfer"

- The Circuit Generator, Alice, has two strings, $\{s_0, s_1\}$
- The Circuit Evaluator, Bob, has a bit b

SMPC Protocols: Garbled Circuit Evaluation

"1-out-of-2 Oblivious Transfer"

- The Circuit Generator, Alice, has two strings, $\{s_0, s_1\}$
- The Circuit Evaluator, Bob, has a bit b
- Alice $\{s_0, s_1\}$ -> Oblivious Transfer

SMPC Protocols: Garbled Circuit Evaluation

"1-out-of-2 Oblivious Transfer"

- The Circuit Generator, Alice, has two strings, $\{s_0, s_1\}$
- The Circuit Evaluator, Bob, has a bit b
- Alice $\{s_0, s_1\}$ -> Oblivious Transfer -> Bob $\{s_b\}$

SMPC Protocols: Garbled Circuit Evaluation

"1-out-of-2 Oblivious Transfer"

- The Circuit Generator, Alice, has two strings, $\{s_0, s_1\}$
- The Circuit Evaluator, Bob, has a bit b
- Alice $\{s_0, s_1\}$ -> Oblivious Transfer -> Bob $\{s_b\}$
- Alice doesn't know which string Bob received
- Bob has the needed string to evaluate his inputs

SMPC Protocols: Garbled Circuit Evaluation

GCE uses a randomized “Garbling Algorithm” $G(\ )$ on a function $f(\ )$ to turn it into three separate functions:

$\boldsymbol{G(f(\ ))}$ -> $\{\boldsymbol{f_e(\ ), f_g(\ ), f_d(\ )}\}$

SMPC Protocols: Garbled Circuit Evaluation

GCE uses a randomized "Garbling Algorithm" $G(\ )$ on a function $f(\ )$ to turn it into three separate functions:

$G(f(\ )) \rightarrow \{f_e(\ ), f_g(\ ), f_d(\ )\}$

- $\boldsymbol{f_e(\ )}$ is an encoding function which allows us to evaluate the "Garbled Inputs": $f_e(i_A) = i_{Ae}$ and $f_e(i_B) = i_{Be}$

SMPC Protocols: Garbled Circuit Evaluation

GCE uses a randomized “Garbling Algorithm” $G(\ )$ on a function $f(\ )$ to turn it into three separate functions:

$G(f(\ ))$ -> $\{f_e(\ ), f_g(\ ), f_d(\ )\}$

- $f_e(\ )$ is an encoding function which allows us to evaluate the “Garbled Inputs”: $f_e(i_A) = i_{Ae}$ and $f_e(i_B) = i_{Be}$
- $\boldsymbol{f_g(\ )}$ is the garbled representation of the original circuit, computes the garbled output: $f_g(i_{Ae}, i_{Be}) = o_g$

SMPC Protocols: Garbled Circuit Evaluation

GCE uses a randomized “Garbling Algorithm” $G(\ )$ on a function $f(\ )$ to turn it into three separate functions:

$G(f(\ ))$ -> $\{f_e(\ ), f_g(\ ), f_d(\ )\}$

- $f_e(\ )$ is an encoding function which allows us to evaluate the “Garbled Inputs”: $f_e(i_A) = i_{Ae}$ and $f_e(i_B) = i_{Be}$
- $f_g(\ )$ is the garbled representation of the original circuit, computes the garbled output: $f_g(i_{Ae}, i_{Be}) = o_g$
- $\boldsymbol{f_d(\ )}$ is a decoding function which allows us to receive the final output of the computation: $f_d(o_g) = o_{final}$

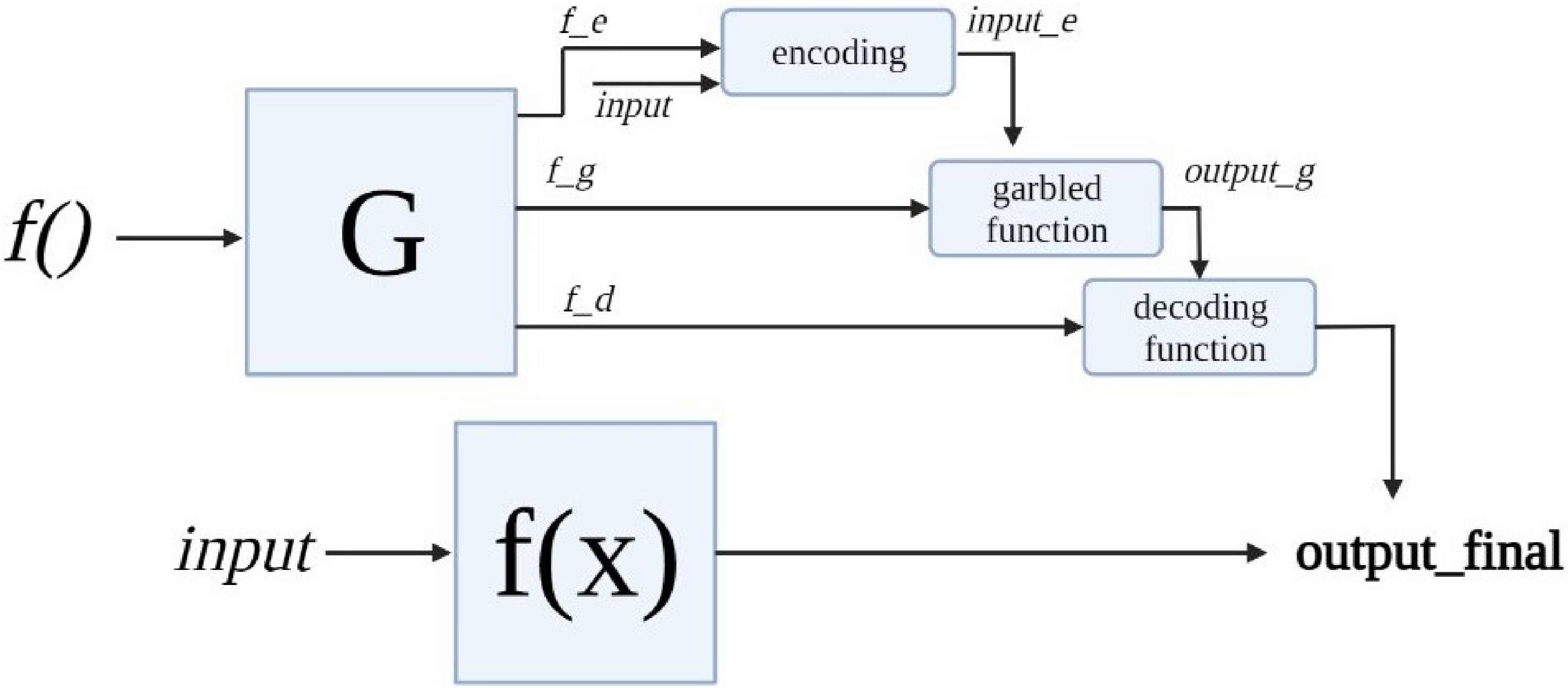
SMPC Protocols: Garbled Circuit Evaluation

GCE uses a randomized “Garbling Algorithm” $G(\ )$ on a function $f(\ )$ to turn it into three separate functions:

$G(f(\ )) \rightarrow \{f_e(\ ), f_g(\ ), f_d(\ )\}$

- $f_e(\ )$ is an encoding function which allows us to evaluate the “Garbled Inputs”: $f_e(i_A) = i_{Ae}$ and $f_e(i_B) = i_{Be}$
- $f_g(\ )$ is the garbled representation of the original circuit, computes the garbled output: $f_g(i_{Ae}, i_{Be}) = o_g$
- $f_d(\ )$ is a decoding function which allows us to receive the final output of the computation: $f_d(o_g) = o_{final}$
- In order to receive the correct result, it’s required that $f(\ ) = (f_e(\ ) \circ f_g(\ ) \circ f_d(\ ))$

SMPC Protocols: Garbled Circuit Evaluation



SMPC Protocols: Garbled Circuit Evaluation

- $G(\ )$ must be randomized
- Both participants Alice and Bob receive $f_e(\ )$ right away
- $f_g(\ )$ returns the encoded result of the boolean circuit
- This allows for control over who receives the result of the computation

Talk Outline

1. INTRODUCTION
    a. What is SMPC?
    b. Implementations of SMPC
    c. GCE, nanoPI, and Conclave
2. BACKGROUND
    a. Security Classifications of SMPC
3. *SMPC PROTOCOLS*
    a. *Garbled Circuit Evaluation*
4. SMPC SCALING
    a. Scaling Issues
    b. nanoPI
    c. Conclave
5. CONCLUSION

SMPC Scaling: Scaling Issues

- SMPC is used on massive amounts of data
    - Medical Data, Financial Data
- Boolean circuits are large scale representations of functions
- SMPC protocols will often generate circuits with billions of gates
- Actively secure SMPC protocols require either linear space or linear rounds of computations
- Modern Actively Secure GCE protocols compute hundreds of thousands of gates per second
- Even powerful GCE protocols requires linear space, scaling poorly with size

SMPC Scaling: Scaling Issues

- Many possible applications of SMPC on resource constrained devices
  - Smart Watches
  - IoT Devices
- Majority of SMPC scaling advancements only made on passively secure protocols
- Actively secure SMPC protocols achieve constant-round time efficiency by trading:
  - |*Circuit*| space

SMPC Scaling: nanoPI

nanoPI

- Based on SMPC protocol WRK
    - Incredibly Efficient
    - Actively secure against n-1 active adversaries
- Developed by fixing space inefficiencies within WRK
- Modifies sub-protocols of WRK to batch operations

SMPC Scaling: nanoPI

- Authenticated Bit Changes:
  - Changes made to batch repeated computations of the Authenticated bit protocol in order to save space
- Authenticated AND Changes:
  - Takes advantage of a predictable pool based cut-and-choose system, freeing up space that would otherwise be needed for some Authenticated AND computations

SMPC Scaling: nanoPI

Memory Budget		20 MB		200 MB		2 GB	
Protocol		nanoPI	WRK	nanoPI	WRK	nanoPI	WRK
Speed (AND/s)	20 Mbps 40 ms	795.03	1.73K	2.73K	2.75K	3.12K	3.23K
	200 Mbps 40ms	825.18	2.76K	6.94K	12.94K	20.94K	22.38K
	2 Gbps <1 ms	20.27K	20.53K	46.66K	46.84K	49.34K	50.64K
Bandwidth (Byte/AND)		505	504	380	504	379	378

SMPC Scaling: Conclave

Conclave

- Query compiler built to efficiently perform SMPC
- Parts of relational analytics queries can be performed insecurely
- Hybrid Datasets
- Conclave computes SMPC steps & unprotected data in parallel with SMPC processes
- Can be done locally in cleartext due to the nature of these operations

SMPC Scaling: Conclave

Mechanics:

- A data analyst writes relational queries for the data as if they had access to all inputs
- Conclave transforms them into a set of local processing steps and modified SMPC steps
- Conclave runs private SMPC steps and local processing steps in parallel
- Conclave is often able to return the results of an SMPC protocol within minutes, even on large scale inputs

SMPC Scaling: Conclave

- Conclave can generate code for GCE and Secret Sharing SMPC protocols
- Conclave is only passively secure
- In testing: Conclave sped up operators such as "join" and "aggregate" to 7 plus times faster when compared to Sharemind, a commercial framework used in applications of SMPC
- These tests were done on hybrid data sets
- Conclave is best suited for SMPC applications relating to research

Talk Outline

1. INTRODUCTION
    a. What is SMPC?
    b. Implementations of SMPC
    c. GCE, nanoPI, and Conclave
2. BACKGROUND
    a. Security Classifications of SMPC
3. *SMPC PROTOCOLS*
    a. *Garbled Circuit Evaluation*
4. SMPC SCALING
    a. Scaling Issues
    b. nanoPI
    c. Conclave
5. CONCLUSION

Conclusion: Partial Solutions

- nanoPI and Conclave are partial solutions
- nanoPI:
  - Kept up with WRK, however still slower in all tests
  - Has the unique ability to run arbitrarily large circuits
- Conclave:
  - 7+ times faster use of important operators in testing
  - Only passively secure, and requires a hybrid data-set to tap into most of its efficiency
  - Perfect for data analytics such as Medical Research

References

Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. 2012. Foundations of Garbled Circuits. In Proceedings of the 2012 ACM Conference on Computer and Communications Security (Raleigh, North Carolina, USA) (CCS '12). Association for Computing Machinery, New York, NY, USA, 784–796. https://doi.org/10.1145/2382196.2382279

Nikolaj Volgushev, Malte Schwarzkopf, Ben Getchell, Mayank Varia, Andrei Lapets, and Azer Bestavros. 2019. Conclave: Secure Multi-Party Computation on Big Data. In Proceedings of the Fourteenth EuroSys Conference 2019 (Dresden, Germany) (EuroSys '19). Association for Computing Machinery, New York, NY, USA, Article 3, 18 pages. https://doi.org/10.1145/3302424.3303982

Ruiyu Zhu, Darion Cassel, Amr Sabry, and Yan Huang. 2018. NANOPI: Extreme-Scale Actively-Secure Multi-Party Computation. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (Toronto, Canada) (CCS '18). Association for Computing Machinery, New York, NY, USA, 862–879. https://doi.org/10.1145/3243734.3243850

Questions?