Senate
A Maliciously Secure MPC Platform for Collaborative Analytics

Rishabh Poddar 1,2, Sukrit Kalra 1, Avishay Yanai 3, Ryan Deng 1, Raluca Ada Popa 1,2, and Joseph M. Hellerstein 1

1 UC Berkeley  2 Opaque Systems  3 VMware Research

USENIX Security 2021
Rich collaborative analytics on shared data

Medical studies  Financial services  Online advertising
Rich collaborative analytics on shared data

Example query: Disease comorbidity

```
SELECT diagnosis, COUNT (*) count
FROM patients P1 U patients P2 U patients P3
WHERE has_cdiff = 'True'
GROUP BY diagnosis ORDER BY count LIMIT 10

[Bater+17]
```
Rich collaborative analytics on shared data

Example query: Disease comorbidity

```
SELECT diagnosis, COUNT (*) count
FROM patients
WHERE has_cdiff = 'True'
GROUP BY diagnosis ORDER BY count LIMIT 10

[Bater+17]
```
Rich collaborative analytics on shared data

**Example query: Disease comorbidity**

```sql
SELECT diagnosis, COUNT(*) count
FROM patients|P1 U patients|P2 U patients|P3
WHERE has_cdiff = 'True'
GROUP BY diagnosis ORDER BY count LIMIT 10
```

[Bater+17]
Rich collaborative analytics on shared data

Example query: Disease comorbidity

SELECT diagnosis, COUNT (*) count
FROM patients|P1 U patients|P2 U patients|P3
WHERE has_cdiff = 'True'
GROUP BY diagnosis ORDER BY count LIMIT 10

[Bater+17]
Rich collaborative analytics on shared data

Example query: Disease comorbidity

```
SELECT diagnosis, COUNT(*) count
FROM patients|P1 U patients|P2 U patients|P3
WHERE has_cdiff = 'True'
GROUP BY diagnosis
ORDER BY count LIMIT 10
```

[Bater+17]
Problem: Privacy

Example query: Disease comorbidity

```
SELECT diagnosis, COUNT (*) count
FROM patients|P1 U patients|P2 U patients|P3
WHERE has_cdiff = 'True'
GROUP BY diagnosis ORDER BY count LIMIT 10
```
Problem: Privacy

- Privacy concerns
- Laws and regulations
- Business competition
Secure multiparty computation (MPC)

- Enables parties to share and compute on encrypted data

[Yao82, GMW87, BGW88]
Secure multiparty computation (MPC)

- Enables parties to share and compute on encrypted data
- No party learns any party’s input beyond the final result

[Yao82, GMW87, BGW88]
Threat model & Desired security guarantees
Threat model & Desired security guarantees

- All parties can provide arbitrary input
Threat model & Desired security guarantees

- All parties can provide arbitrary input
- Protocol should be secure even if all other parties collude (dishonest majority)
Threat model & Desired security guarantees

• All parties can provide arbitrary input
• Protocol should be secure even if all other parties collude (dishonest majority)
• Protocol should be secure even if the adversary deviates from the protocol (malicious security)
Current state of the art: Monolithic circuits
e.g. [WRK17]

All parties need to execute a *monolithic* cryptographic computation *together* to evaluate the desired function
Can we “decompose” monolithic MPC?
Can we “decompose” monolithic MPC?

Monolithic MPC

Plaintext execution
Can we “decompose” monolithic MPC?

Monolithic MPC

Plaintext execution

Can we “decompose” monolithic MPC?
Can we “decompose” monolithic MPC?

Monolithic MPC

Plaintext execution

Parallel operations
Can we “decompose” monolithic MPC?
Can we “decompose” monolithic MPC?

Monolithic MPC

Drawbacks

• No parallelism across parties
Can we “decompose” monolithic MPC?

**Drawbacks**

- No parallelism across parties
- No local plaintext computation
Can we “decompose” monolithic MPC?

Monolithic MPC

Drawbacks

- No parallelism across parties
- No local plaintext computation
- All parties jointly execute every sub-computation, even if it doesn’t directly involve their input
Can we “decompose” monolithic MPC?

How to decompose without compromising security?
Challenges in MPC decomposition

Strawman MPC decomposition

SELECT

JOIN

JOIN

JOIN

x_1

x_2

x_3

x_4

P1

P2

P3

P4
Challenges in MPC decomposition

Strawman MPC decomposition

Challenges

1. Decomposition reveals intermediate results to the adversary
Challenges in MPC decomposition

Strawman MPC decomposition

Challenges

1. Decomposition reveals intermediate results to the adversary

2. Adversary can provide invalid intermediate inputs
Challenges in MPC decomposition

Challenges

1. Decomposition reveals intermediate results to the adversary

2. Adversary can provide *invalid* intermediate inputs

3. Adversary can provide *inconsistent* inputs
Key technique: Secure MPC decomposition

Senate's MPC protocol

Challenges

1. Decomposition reveals intermediate results to the adversary
2. Adversary can provide invalid intermediate inputs
3. Adversary can provide inconsistent inputs
Key technique: Secure MPC decomposition

Senate’s MPC protocol

Challenges

1. Decomposition reveals intermediate results to the adversary
2. Adversary can provide invalid intermediate inputs
3. Adversary can provide inconsistent inputs
Key technique: Secure MPC decomposition

Senate’s MPC protocol

Secure MPC decomposition even in the presence of malicious adversaries
Key technique: Secure MPC decomposition

Senate's MPC protocol

Secure MPC decomposition even in the presence of malicious adversaries

- Enables local computation
Key technique: Secure MPC decomposition

Secure MPC decomposition even in the presence of malicious adversaries

- Enables local computation
- Sub-computations involving different parties can proceed in parallel
Key technique: Secure MPC decomposition

Secure MPC decomposition even in the presence of malicious adversaries

- Enables local computation
- Sub-computations involving different parties can proceed in parallel
- Sub-computations involve only required subset of parties
Senate’s contributions

Secure MPC decomposition protocol
- Solders sub-computations together for security of intermediate inputs
- Enforces integrity of sub-computations
- Formalizes class of admissible decompositions

Designing efficient circuits for SQL operations
- New Boolean circuit primitives for multiparty operations: $m$-SI, $m$-SU, $m$-Sort, Verifiers
- Realizing SQL operators using the circuit primitives: joins, group by, order by, filters

Executing queries using Senate’s MPC decomposition protocol
- New algorithms for planning the representation and execution of SQL queries
- Cost model for determining the optimal decomposition
Senate’s contributions

Secure MPC decomposition protocol
• *Solders* sub-computations together for security of intermediate inputs
• Enforces integrity of sub-computations
• Formalizes class of *admissible* decompositions

Designing efficient circuits for SQL operations
• New Boolean circuit primitives for multiparty operations: $m$-SI, $m$-SU, $m$-Sort, Verifiers
• Realizing SQL operators using the circuit primitives: joins, group by, order by, filters
Senate’s contributions

Secure MPC decomposition protocol
• *Solders* sub-computations together for security of intermediate inputs
• Enforces integrity of sub-computations
• Formalizes class of admissible decompositions

Designing efficient circuits for SQL operations
• New Boolean circuit primitives for multiparty operations: \( m\)-SI, \( m\)-SU, \( m\)-Sort, Verifiers
• Realizing SQL operators using the circuit primitives: joins, group by, order by, filters

Executing queries using Senate’s MPC decomposition protocol
• New algorithms for planning the representation and execution of SQL queries
• Cost model for determining the optimal decomposition
**Senate’s contributions**

Secure MPC decomposition protocol
- *Solders* sub-computations together for security of intermediate inputs
- Enforces integrity of sub-computations
- Formalizes class of *admissible* decompositions

Designing efficient circuits for SQL operations
- New Boolean circuit primitives for multiparty operations: $m$-SI, $m$-SU, $m$-Sort, Verifiers
- Realizing SQL operators using the circuit primitives: joins, group by, order by, filters

Executing queries using Senate’s MPC decomposition protocol
- New algorithms for planning the representation and execution of SQL queries
- Cost model for determining the optimal decomposition
1. Preventing leakage of intermediate values
1 Preventing leakage of intermediate values
Preventing leakage of intermediate values

Key technique:

• New lightweight “soldering” technique for WRK [WRK17] circuits that masks intermediate values
Preventing leakage of intermediate values

Key technique:
- New lightweight “soldering” technique for WRK [WRK17] circuits that masks intermediate values
- Set of parties in first circuit can be a subset of second
Ensuring validity of intermediate inputs
Ensuring validity of intermediate inputs

Key technique:

- Parent circuit verifies the validity of intermediate inputs
Ensuring validity of intermediate inputs

Key technique:

- Parent circuit verifies the validity of intermediate inputs
- Formalize the class of *admissible* decompositions — every sub-computation must be easily invertible
Ensuring consistency of inputs
Ensuring consistency of inputs

Key technique:

- Restrict admissible decompositions to trees and not graphs
Evaluation Highlights
Performance on TPC-H analytics benchmark

Industry standard benchmark for analytics queries

- Comprises a rich set of 22 complex queries on data split across 8 tables
- No notion of multiple “parties”, so we assume one table per party
Performance on TPC-H analytics benchmark

Industry standard benchmark for analytics queries
- Comprises a rich set of 22 complex queries on data split across 8 tables
- No notion of multiple “parties”, so we assume one table per party

Methodology
- r5.12x large AWS instances in LAN and WAN settings
- Baseline: AGMPC framework (implements monolithic WRK)
Performance on TPC-H analytics benchmark

Senate supports 13 of 22 queries, up to 145x faster than the baseline
Summary

Senate is an MPC platform for securely executing collaborative analytical queries in the presence of malicious adversaries.

Improves performance over the state of the art by up to 145x:
- Protocol for secure MPC decomposition
- Efficient query planning and execution algorithms based on a cost model
Summary

Senate is an MPC platform for securely executing collaborative analytical queries in the presence of malicious adversaries.

Improves performance over the state of the art by up to 145x:

- Protocol for secure MPC decomposition
- Efficient query planning and execution algorithms based on a cost model

Thanks!

rishabhp@berkeley.edu