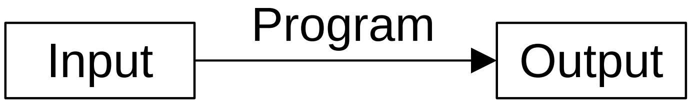
VDORAM: Towards a Random Access Machine with Both Public Verifiability and Distributed Obliviousness

Huayi Qi[1,2], Minghui Xu[1], Xiaohua Jia[3], Xiuzhen Cheng[1]

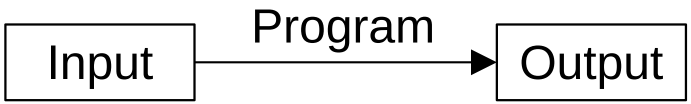
[1] Shandong University
[2] Tsinghua University
[3] City University of Hong Kong

Network and Distributed System Security (NDSS) Symposium 2026

Background

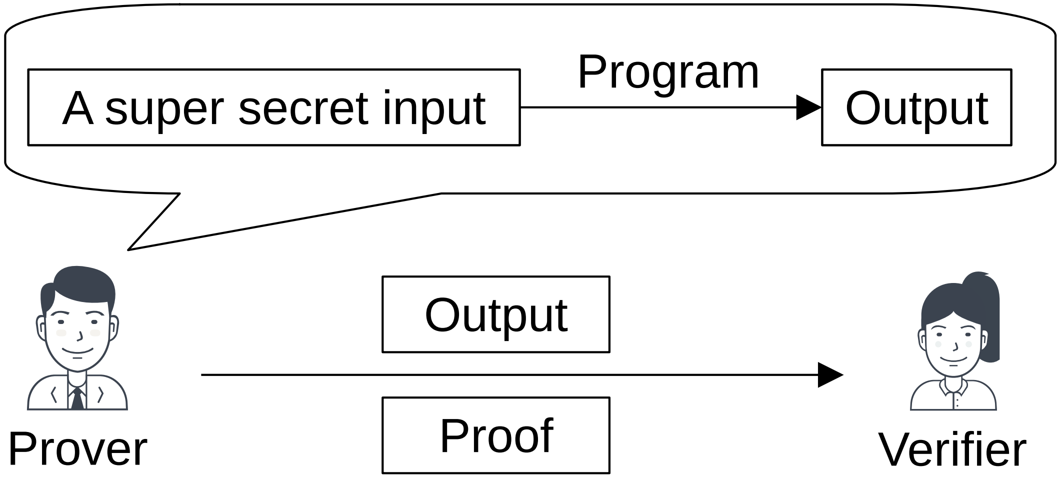


Background



Non-interactive

zero-knowledge proof

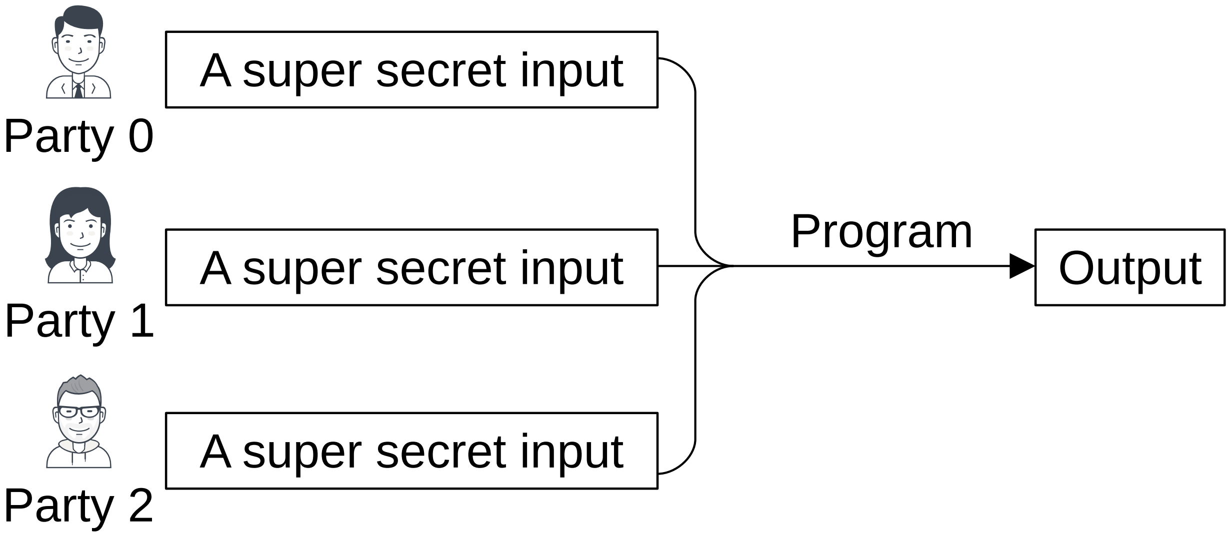


- Typically, one prover
- Prover knows all secrets

- Can be anyone
- Can verify anytime

Background

Multi-party computation



Secrets are not leaked if:

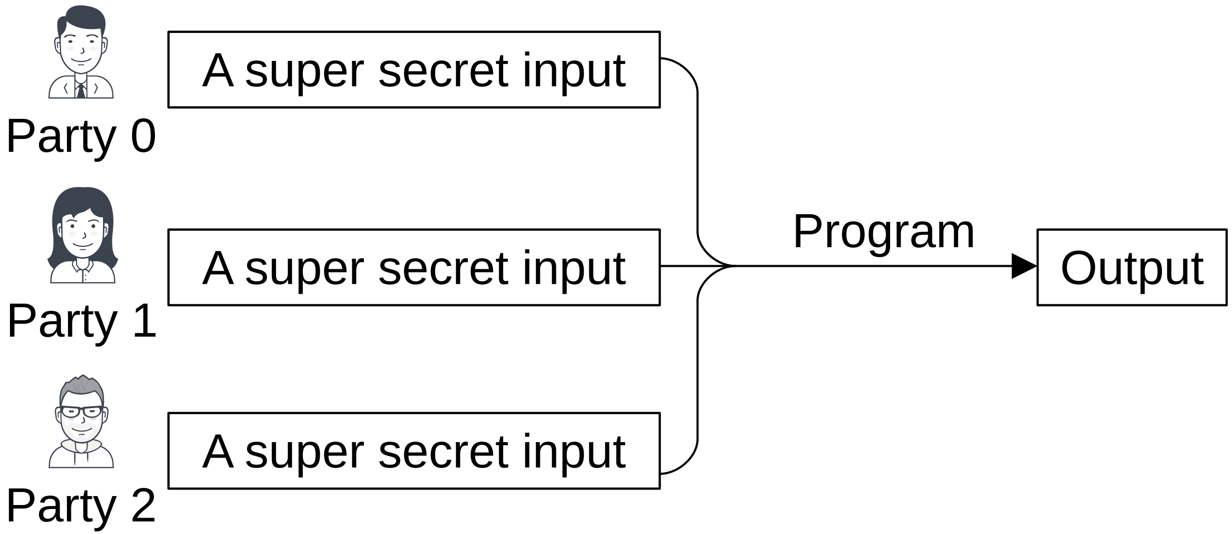
- Assume there are *n* - *t* good guys
- *n* - *t* is at least 1 (e.g., GMW)

Output is correct if additionally:

- The protocol is malicious secure

Background

Multi-party computation



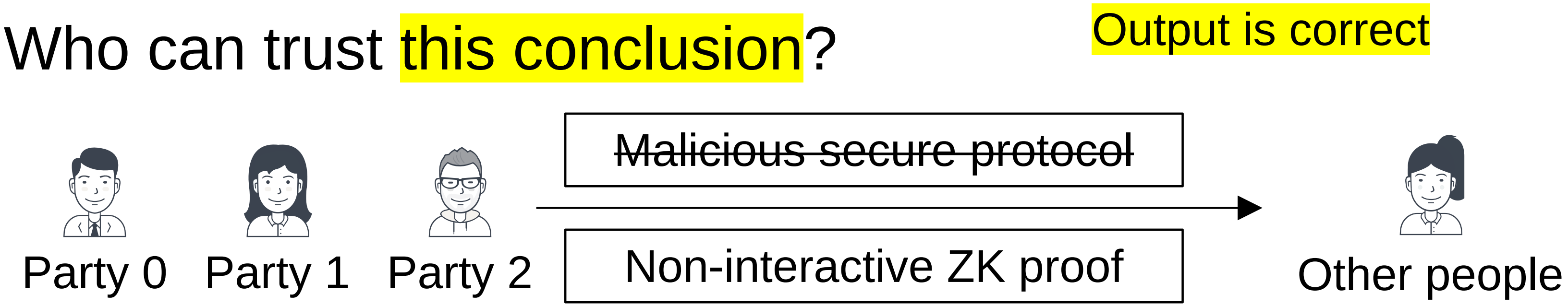
Who can trust this conclusion?

Output is correct if additionally:

- The protocol is malicious secure

Background

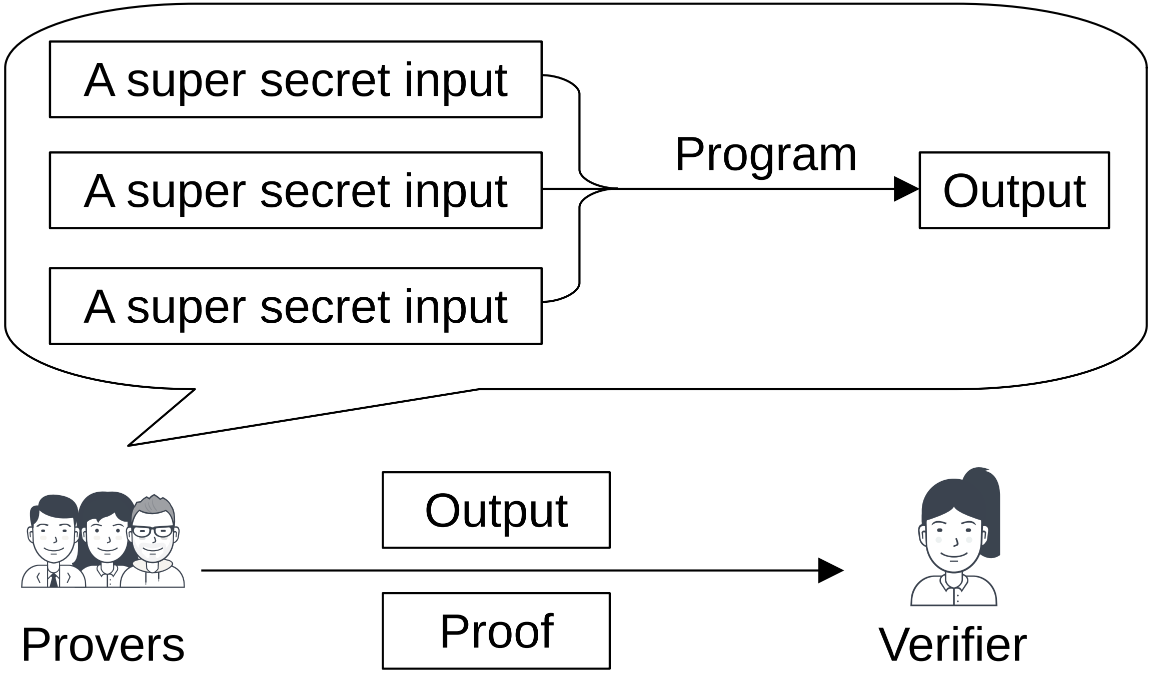
Who can trust this conclusion?



Background

PA-MPC:

Public auditable MPC



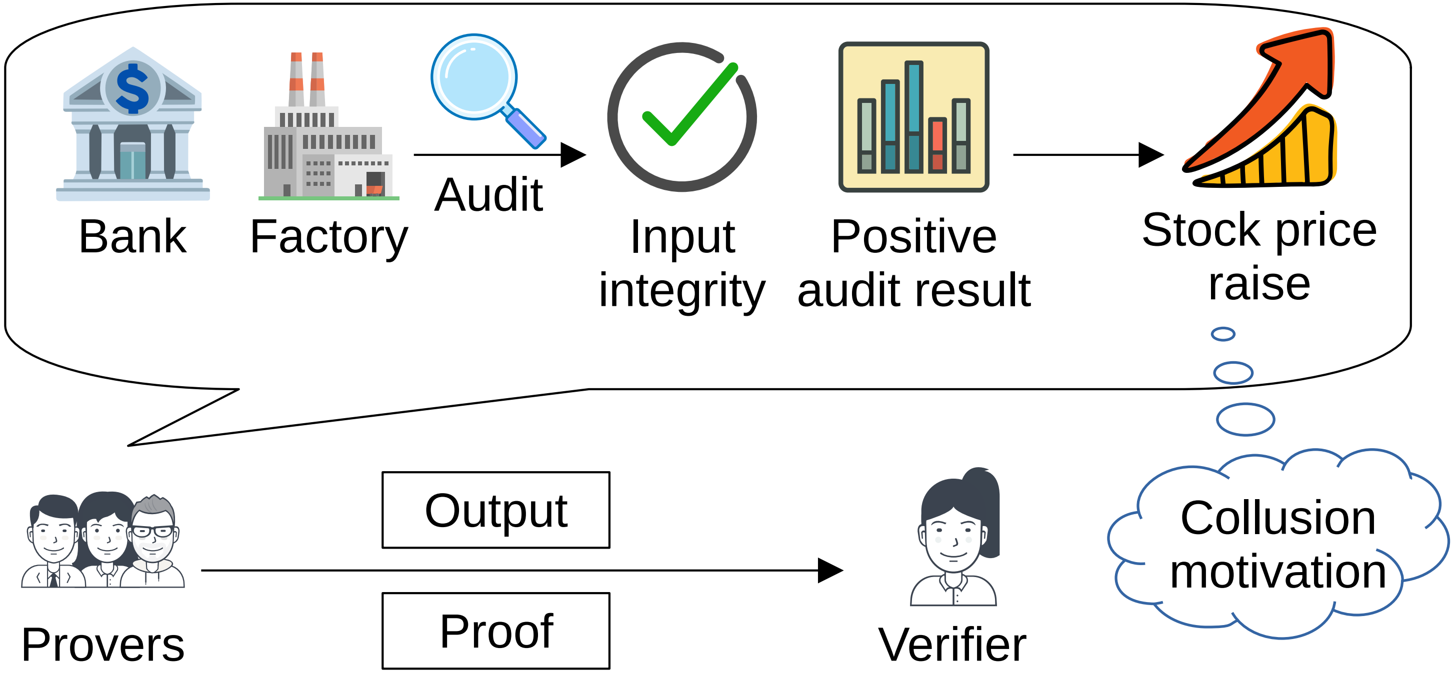
- Multiple provers
- No prover knows all secrets
- Collude if faking the output is possible; protect secrets otherwise

- Can be anyone
- Can verify anytime
- Do not have the *n* - *t* assumption

Background

PA-MPC:

Public auditable MPC



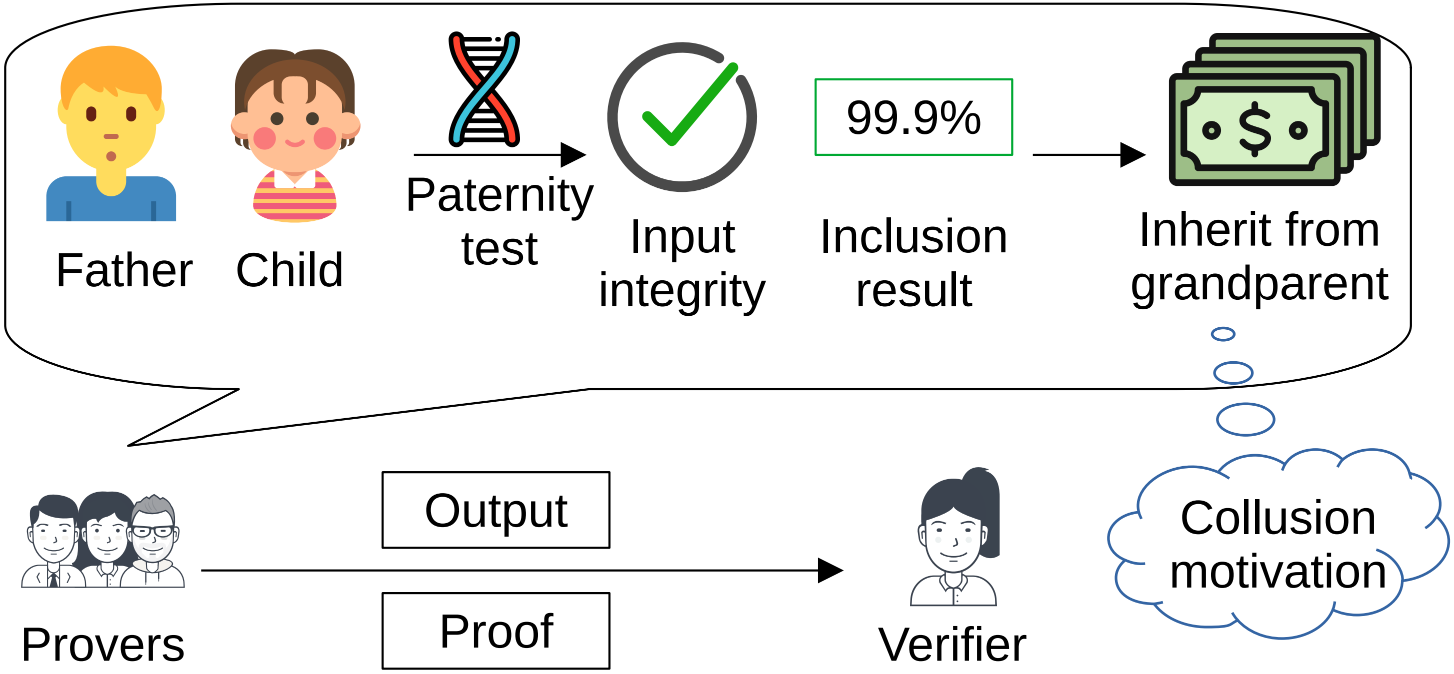
- Multiple provers
- No prover knows all secrets
- Collude if faking the output is possible; protect secrets otherwise

- Can be anyone
- Can verify anytime
- Do not have the n - t assumption

Background

PA-MPC:

Public auditable MPC



- Multiple provers
- No prover knows all secrets
- Collude if faking the output is possible; protect secrets otherwise

- Can be anyone
- Can verify anytime
- Do not have the *n* - *t* assumption

The current status

PA-MPC:

Public auditable MPC

↑ *Partially* implements

Collaborative zkSNARKs [1]

[1] Ozdemir, Alex, and Dan Boneh. "Experimenting with collaborative zk-SNARKs: Zero-Knowledge proofs for distributed secrets." 31st USENIX Security Symposium (USENIX Security 22). 2022.

But there is a pattern mismatch

PA-MPC:

Public auditable MPC

↑ *Partially* implements

Collaborative zkSNARKs [1]

Computation pattern

$$y_1, y_2, \ldots, y_n = f(x_1, x_2, \ldots, x_n)$$

x: input (private)
y: output (public)

Constraint pattern

$$(x, w_1, w_2, \ldots, w_n) \in R$$

x: instance (public)
w: witness (private)

[1] Ozdemir, Alex, and Dan Boneh. "Experimenting with collaborative zk-SNARKs: Zero-Knowledge proofs for distributed secrets." 31st USENIX Security Symposium (USENIX Security 22). 2022.

But there is a pattern mismatch

PA-MPC:

Public auditable MPC

Partially

implements

Collaborative zkSNARKs [1]

Computation pattern

$$y_1, y_2, \dots, y_n = f(x_1, x_2, \dots, x_n)$$

x: input (private)
y: output (public)

F: linear and non-linear computations

Constraint pattern

$$(x, w_1, w_2, \dots, w_n) \in R$$

x: instance (public)
w: witness (private)

Elements $\in \mathbb{F}_p$

R: + and ·
$a \cdot b = c$
$c + d = e$

[1] Ozdemir, Alex, and Dan Boneh. "Experimenting with collaborative zk-SNARKs: Zero-Knowledge proofs for distributed secrets." 31st USENIX Security Symposium (USENIX Security 22). 2022.

Map the MPC computations with ZKP?

PA-MPC:

Public auditable MPC

Partially

implements

Collaborative zkSNARKs [1]

Computation pattern

$$y_1, y_2, \ldots, y_n = f(x_1, x_2, \ldots, x_n)$$

F: linear and non-linear computations

x: input (private)
y: output (public)

If a compiler exists [1]

Constraint pattern

$$(x, w_1, w_2, \ldots, w_n) \in R$$

R: + and ·
$a \cdot b = c$
$c + d = e$

x: instance (public)
w: witness (private)

[1] Ozdemir, Alex, and Dan Boneh. "Experimenting with collaborative zk-SNARKs: Zero-Knowledge proofs for distributed secrets." 31st USENIX Security Symposium (USENIX Security 22). 2022.

Not that easy...

Is it a pure engineering problem?

Computation pattern

$$y_1, y_2, \ldots, y_n = f(x_1, x_2, \ldots, x_n)$$

x: input (private)
y: output (public)

F: linear and non-linear computations

If a compiler exists [1]

Constraint pattern

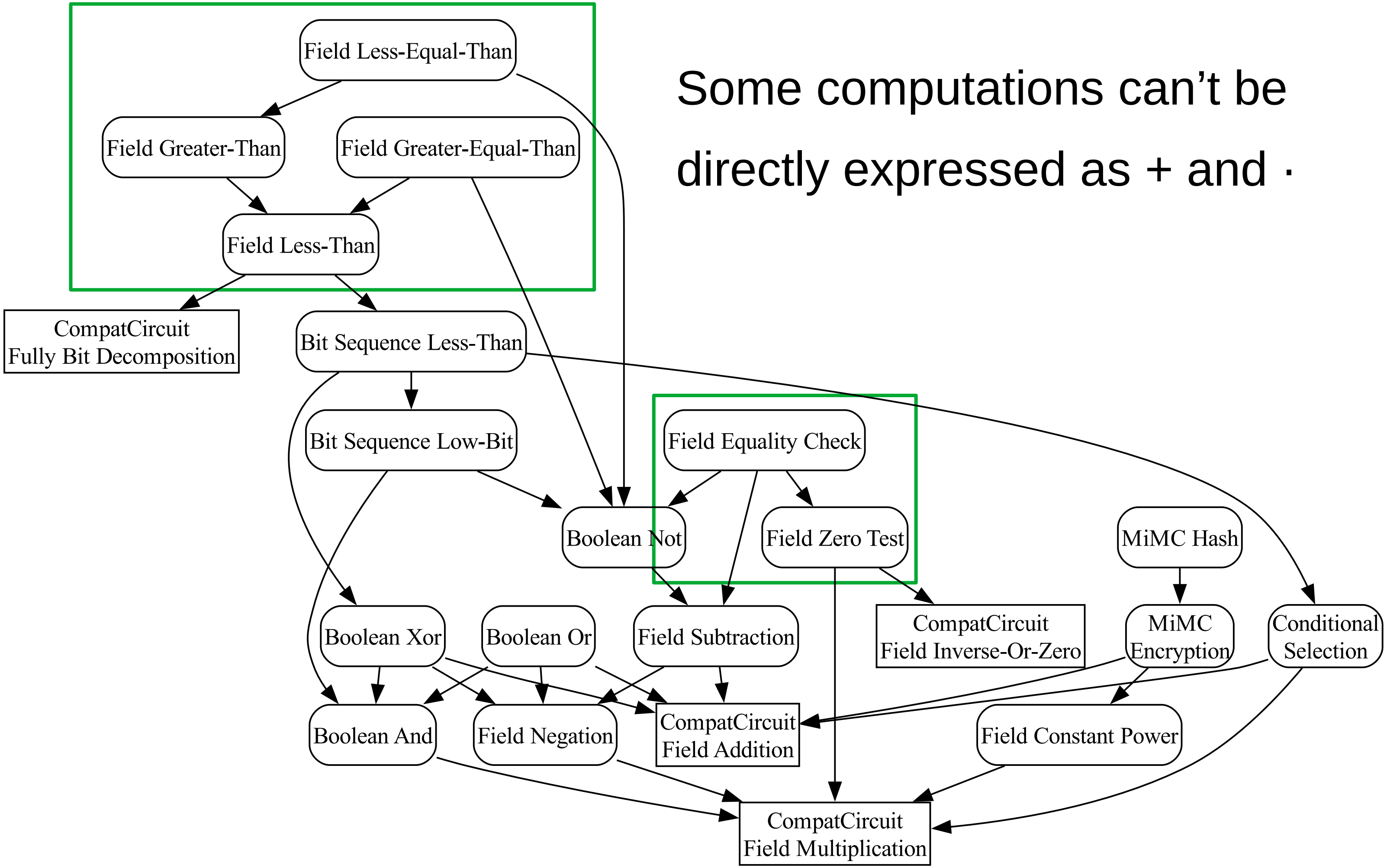
$$(x, w_1, w_2, \ldots, w_n) \in R$$

x: instance (public)
w: witness (private)

R: + and ·
$a \cdot b = c$
$c + d = e$

[1] Ozdemir, Alex, and Dan Boneh. "Experimenting with collaborative zk-SNARKs: Zero-Knowledge proofs for distributed secrets." 31st USENIX Security Symposium (USENIX Security 22). 2022.

Analyze the most frequently used computations



Some computations can't be directly expressed as + and ·

How to utilize non-linear MPC to build constraint?

MPC made for ZKP

- We need an MPC protocol designs in $\mathbb{F}_p$
- Supports linear and non-linear computations
- And outputs *all* values needed to build a ZKP constraint

Secure comparison

c = (a < b ? 1 : 0)

What we would do in a pure MPC? [1]

1. compute *d* = *a* – *b* and preserve the overflow

2. extract the underflow bit of *d*

$$\begin{array}{r} 0b0\underline{0010} \\ -\ 0b0\underline{0111} \\ \text{-------------} \\ 0b1\underline{1011} \end{array}$$

Example:

Data $\in [0, 2^l - 1] (l = 4)$

We compute in $\mathbb{Z}_{2^{l+1}}$

and extract the first bit

This method does not hold in ZKP

[1] Damgård, Ivan, et al. "New primitives for actively-secure MPC over rings with applications to private machine learning." 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019.

Secure comparison

c = (a < b ? 1 : 0)

What we would do in a pure MPC? [1]

1. compute *d* = *a* – *b* and preserve the overflow
2. extract the underflow bit of *d*

$$
\begin{array}{r}
0b0\underline{0010} \\
-\ 0b0\underline{0111} \\
\text{-------------} \\
0b1\underline{1011}
\end{array}
$$

Example:

Data $\in [0, 2^l - 1] (l = 4)$

We compute in $\mathbb{Z}_{2^{l+1}}$

and extract the first bit

This method does not hold in ZKP because

1. underflow bit <==> *a* < *b* only holds in ring $\mathbb{Z}_{2^l}$, not in field $\mathbb{F}_p$

2. we can only build + and · constraints in field $\mathbb{F}_p$, so can't ensure *d* = *a* – *b* holds *in a ring*

3. we can't build a constraint of "*c* is the first bit of *d*" without knowing the whole bit sequence

Secure comparison from bit decomposition

c = (a < b ? 1 : 0)

To enforce a valid ZKP constraint, the MPC protocol must

1. Extract *all* bits of a in field $\mathbb{F}_p$
2. Extract *all* bits of b in field $\mathbb{F}_p$

Constraints: $a = \sum_{i=0}^{l-1} a_i \cdot 2^i$ in $\mathbb{F}_p$; $\{a_i\} < \{p_i\}$ in bit sequences

$\sum_{i=0}^{l-1} a_i \cdot 2^i < p$ in integer

3. Compare the bit sequences by simulating boolean operations in field $\mathbb{F}_p$

AND (x, y) = x · y

NOT (x) = 1 – x

XOR (x, y) = x + y – 2·x·y

OR(x, y) = x + y – x·y

Constraints: + and ·

Secure comparison from bit decomposition

Extract (decompose) *all* bits in field $\mathbb{F}_p$

This means extract l bits where $l < \log_2 p$ does not hold $(l = \lceil \log_2 p \rceil > \log_2 p)$

- MP-SPDZ [1]: support MPC in finite field but only if data bit length $\leqslant \lceil \log_2 p \rceil - 2$
- edaBit [2]: yes as long as the bit length $< \log_2 p$ to avoid overflow

[1] Keller, Marcel. "MP-SPDZ: A versatile framework for multi-party computation." Proceedings of the 2020 ACM SIGSAC conference on computer and communications security. 2020.
[2] Escudero, Daniel, et al. "Improved primitives for MPC over mixed arithmetic-binary circuits." Annual international cryptology conference. Cham: Springer International Publishing, 2020.

Secure comparison from bit decomposition

Extract (decompose) *all* bits in field $\mathbb{F}_p$

This means extract l bits where $l < \log_2 p$ does not hold $(l = \lceil \log_2 p \rceil > \log_2 p)$

- MP-SPDZ [1]: support MPC in finite field but only if data bit length $\leqslant \lceil \log_2 p \rceil - 2$
- edaBit [2]: yes as long as the bit length $< \log_2 p$ to avoid overflow

- [3]: yes
- Ours: an improved version from [3]

[3] Damgård, Ivan, et al. "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation." Theory of Cryptography Conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

Equality test

c = (a == b ? 1 : 0)

MPC equality test in field:

$d = a - b$

$e = d^{p-2}$ we will get $e = 0$ if $d = 0$

$c = d \cdot e$

But how to build the constraint?

- [1]: prepare $r = (a - b)^{-1}$ *if* $a = b$

  a random non-zero *otherwise*

[1] Delpech de Saint Guilhem, Cyprien, et al. "Efficient proof of RAM programs from any public-coin zero-knowledge system." International Conference on Security and Cryptography for Networks. Cham: Springer International Publishing, 2022.

Equality test

c = (a == b ? 1 : 0)

MPC equality test in field:

$d = a - b$

$e = d^{p-2}$ we will get $e = 0$ if $d = 0$

$c = d \cdot e$

But how to build the constraint?

- [1]: prepare $r = (a-b)^{-1}$ *if* $a = b$

  a random non-zero *otherwise*

- [2]: $c + d \cdot e - 1 = 0$ and $c \cdot d = 0$

- Ours: $c \cdot (c \cdot d - 1) = 0$

  and $d \cdot (c \cdot d - 1) = 0$

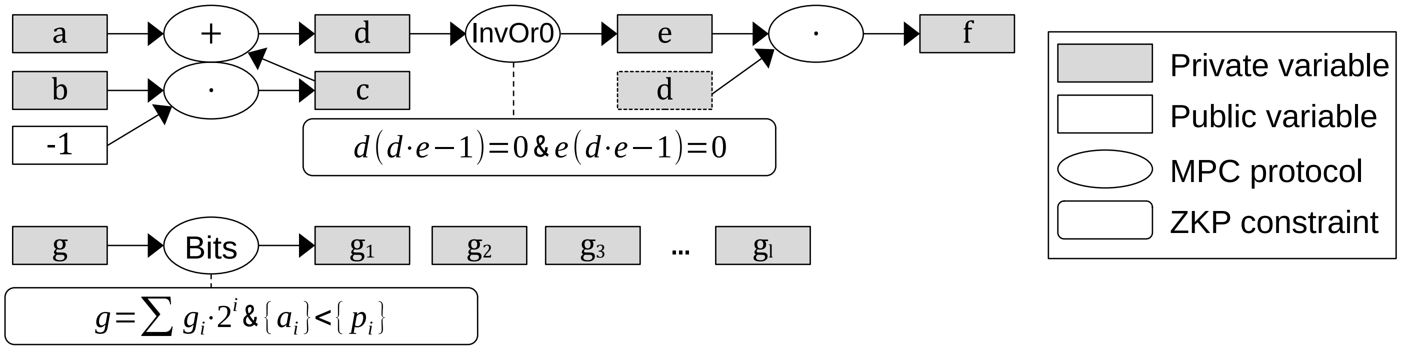
[2] iden3. circomlib: Library of basic circuits for circom. https://github. com/iden3/circomlib

Build a unified representation for MPC and ZKP

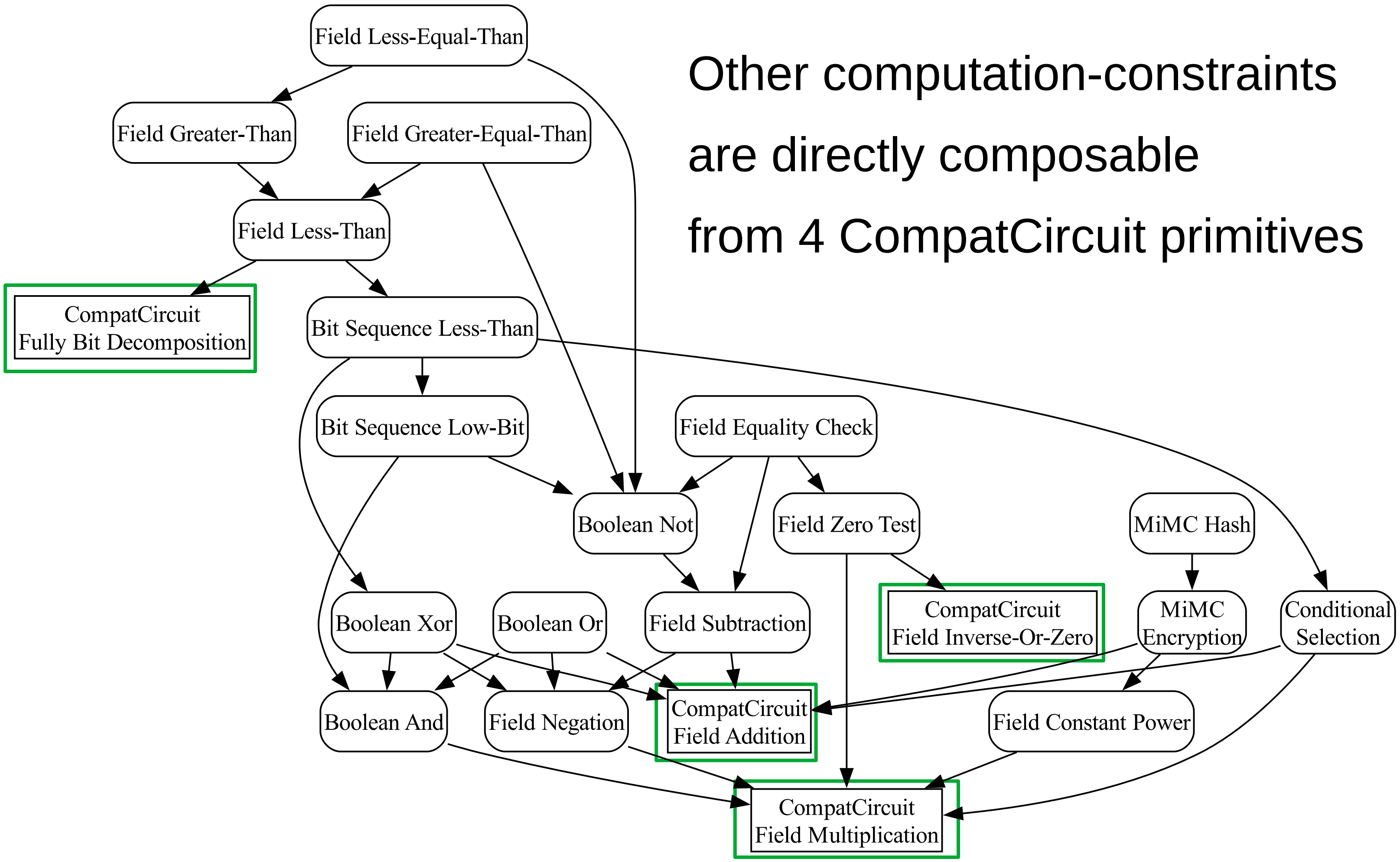
- It is nearly infeasible to write an MPC program *and* a ZKP constraint, followed by manually mapping them

Build a unified representation for MPC and ZKP

- It is nearly infeasible to write an MPC program *and* a ZKP constraint, followed by manually mapping them
- We define a new kind of circuit, CompatCircuit
- It represents the computation and constraint *once*
- It looks similar to the arithmetic circuit but contains non-linear operations
- And *compatible* with both MPC and ZKP purpose



Build a unified representation for MPC and ZKP



Other computation-constraints

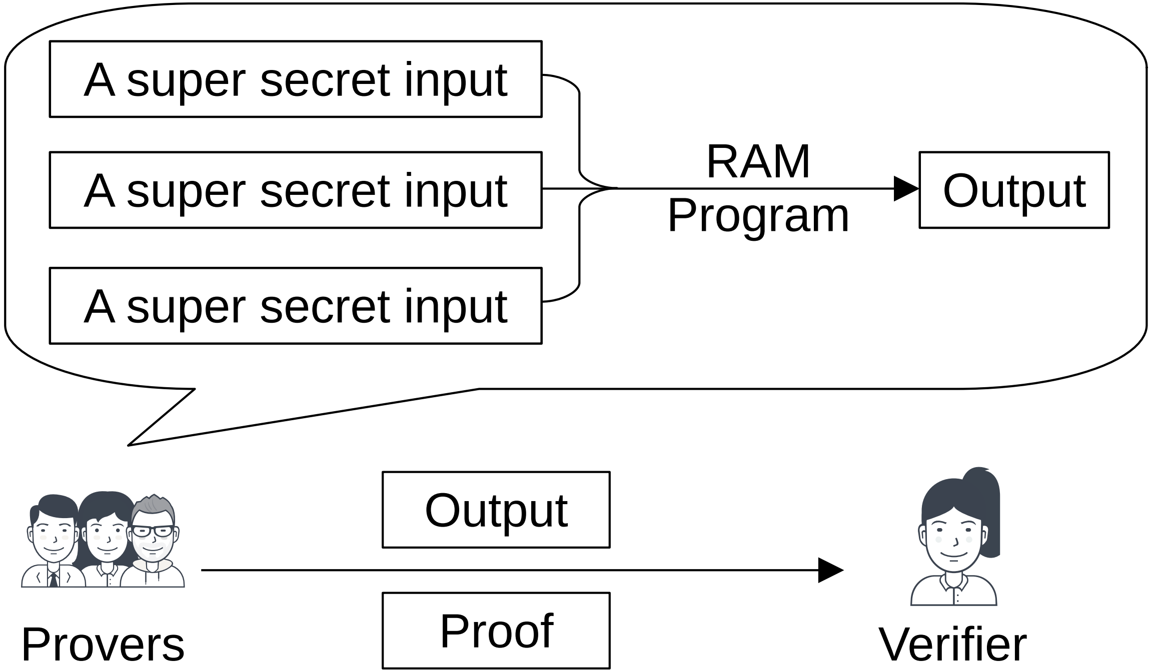
are directly composable

from 4 CompatCircuit primitives

Build a RAM using CompatCircuit

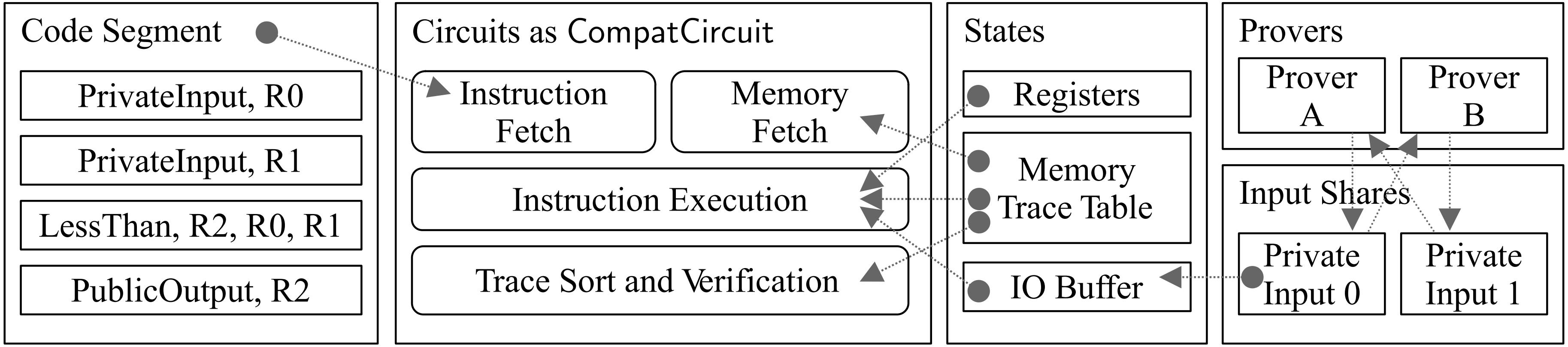
Ours: VDORAM

- vRAM (zkVM):
  public verifiability
- DORAM (RAM-MPC):
  distributed obliviousness



Note: We assume M = Machine in RAM.
There are also some works named DORAM focusing on M = Memory.

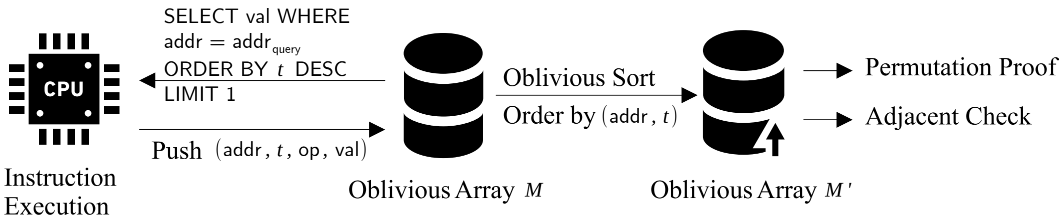
The VDORAM architecture



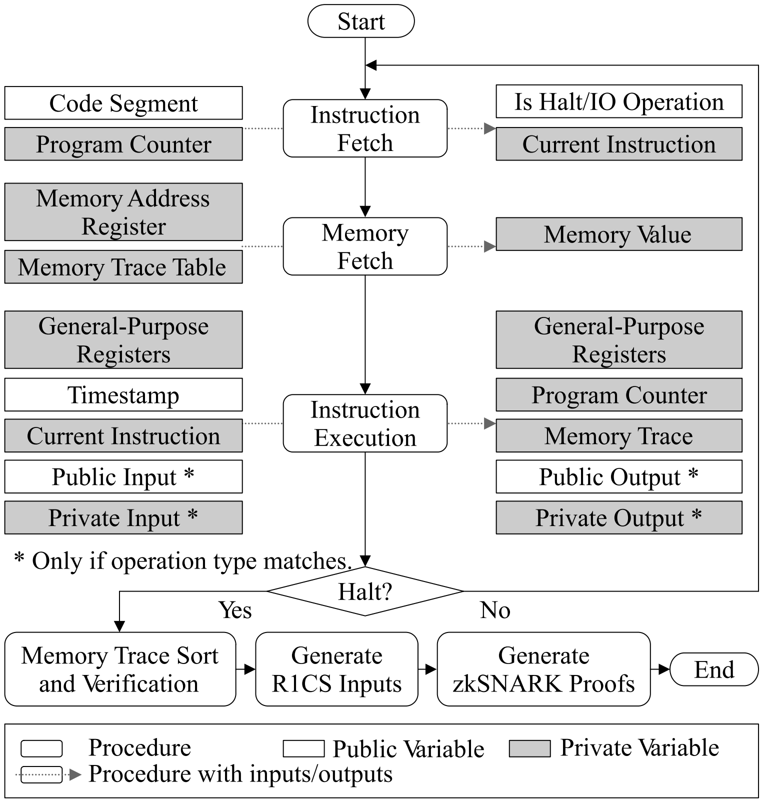
- Circuits are implemented as CompatCircuit
- Allows programmers write a MPC + ZKP program using a register-based instruction set architecture

The VDORAM workflow

- Provers run VDORAM *blindly* like a DORAM (RAM-MPC)
- Provers generate the memory integrity proof like a vRAM (zkVM)



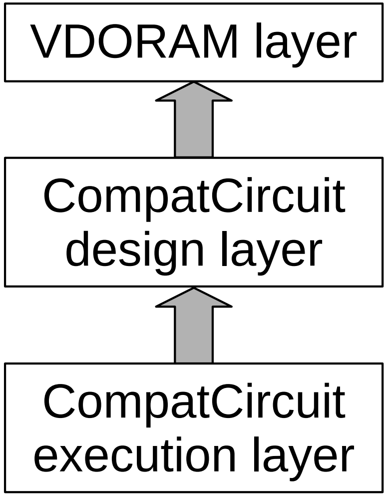
Memory integrity



VDORAM workflow

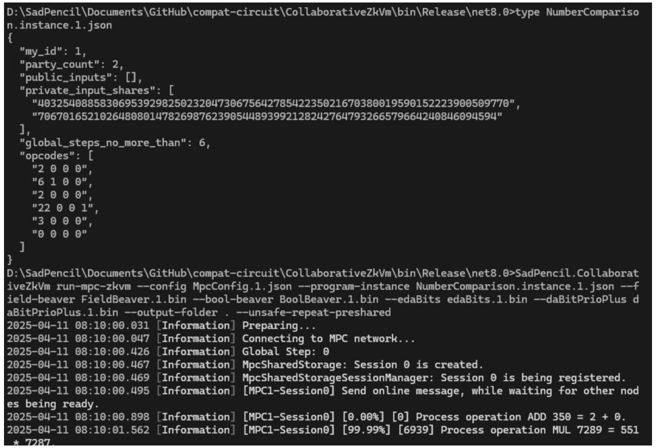
Implementation

- C#, .NET 8. ~ 15,000 lines of codes
- Also integrates collaborative-zksnarks project
- https://github.com/BDS-SDU/vdoram-artifacts




```csharp
public class DemoCircuit : ICircuitBoardGenerator {
    public CircuitBoard GetCircuitBoard() {
        CircuitBoard cb = new();

        // Private inputs
        Wire a = Wire.NewPrivateInputWire("a");
        Wire b = Wire.NewPrivateInputWire("b");
        cb.AddWires(a, b);

        // Comparison gadget
        GadgetInstance g = new FieldLessThanGadget()
            .ApplyGadget([a, b], "a_lt_b");
        g.Save(cb);

        // Public output
        g.OutputWires[0].Name = "out_lt";
        g.OutputWires[0].IsPublicOutput = true;

        return cb;
    }
}
```

CompatCircuit design layer

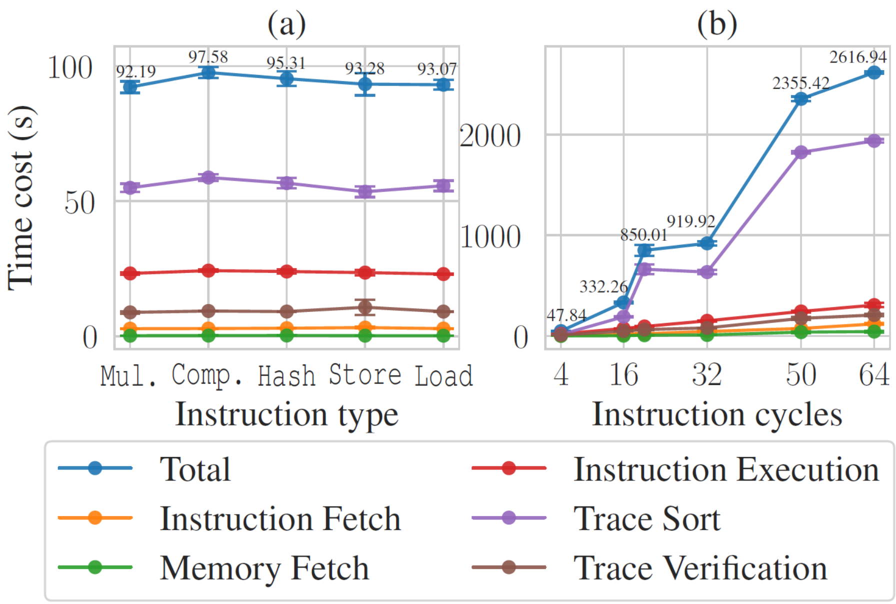


```
D:\SadPencil\Documents\GitHub\compat-circuit\CollaborativeZkVm\bin\Release\net8.0>type NumberCompariso
n.instance.1.json
{
  "my_id": 1,
  "party_count": 2,
  "public_inputs": [],
  "private_input_shares": [
    "4032540885830695392982502320473067564278542235021670380019590152223900509770",
    "70670165210264808014782698762390544893992128242764793266579664240846094594"
  ],
  "global_steps_no_more_than": 6,
  "opcodes": [
    "2 0 0 0",
    "6 1 0 0",
    "2 0 0 0",
    "22 0 0 1",
    "3 0 0 0",
    "0 0 0 0"
  ]
}
D:\SadPencil\Documents\GitHub\compat-circuit\CollaborativeZkVm\bin\Release\net8.0>SadPencil.Collaborat
iveZkVm run-mpc-zkvm --config MpcConfig.1.json --program-instance NumberComparison.instance.1.json --f
ield-beaver FieldBeaver.1.bin --bool-beaver BoolBeaver.1.bin --edaBits edaBits.1.bin --daBitPrioPlus d
aBitPrioPlus.1.bin --output-folder . --unsafe-repeat-preshared
2025-04-11 08:10:00.031 [Information] Preparing...
2025-04-11 08:10:00.047 [Information] Connecting to MPC network...
2025-04-11 08:10:00.426 [Information] Global Step: 0
2025-04-11 08:10:00.467 [Information] MpcSharedStorage: Session 0 is created.
2025-04-11 08:10:00.469 [Information] MpcSharedStorageSessionManager: Session 0 is being registered.
2025-04-11 08:10:00.495 [Information] [MPC1-Session0] Send online message, while waiting for other nod
es being ready.
2025-04-11 08:10:00.898 [Information] [MPC1-Session0] [0.00%] [0] Process operation ADD 350 = 2 + 0.
2025-04-11 08:10:01.562 [Information] [MPC1-Session0] [99.99%] [6939] Process operation MUL 7289 = 551
 * 7287.
```

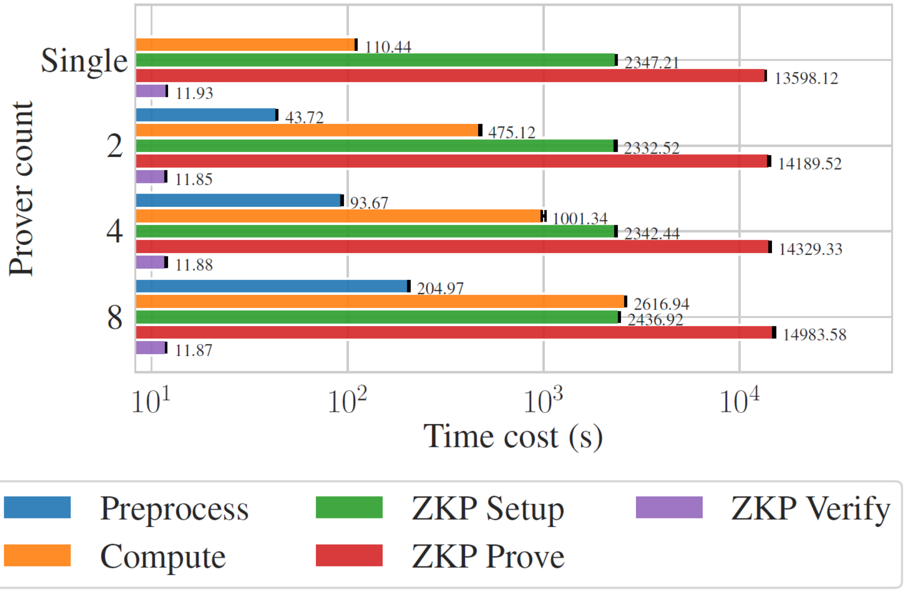
VDORAM screenshot

Evaluation

- Denote *m* parties. Run experiments in a server with 4*m* CPU cores, 4*m* GB RAM, 20*m* GB of SSD. Local network.

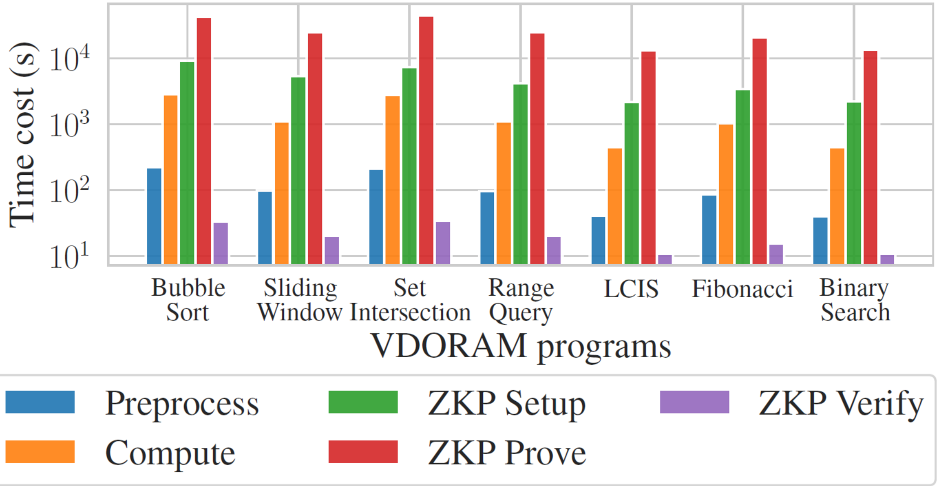


Sorting the memory trace costs the most



The total time cost remains moderate when party count grows

Evaluation



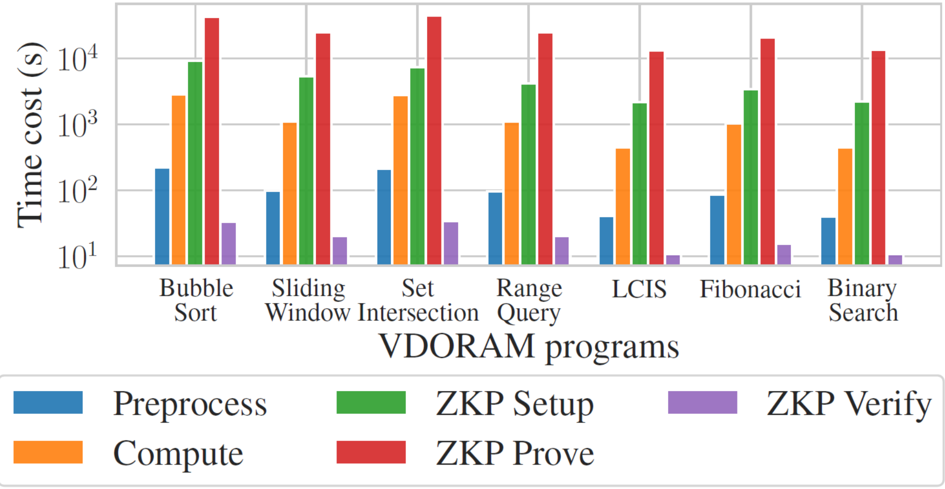
VDORAM running example programs

TABLE III
COMPARISON OF VDORAM WITH PRIOR WORKS

RAM System	Number of Parties	Public Verifiability	Distributed Obliviousness
Jolt [4]	1 prover	Yes	No
SP1 [61]	1 prover	Yes	No
Wang et al. [74]	2 parties	No	Yes
VDORAM (ours)	**$\geq$ 1 provers**	**Yes**	**Yes**

VDORAM achieves V+DO

Evaluation

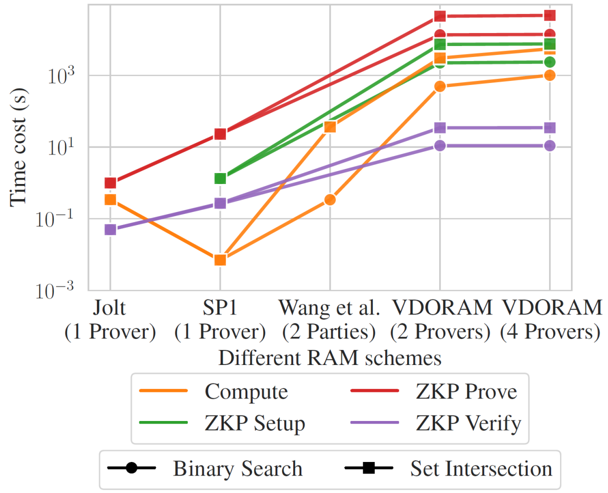


VDORAM running example programs

TABLE III
COMPARISON OF VDORAM WITH PRIOR WORKS

RAM System	Number of Parties	Public Verifiability	Distributed Obliviousness
Jolt [4]	1 prover	Yes	No
SP1 [61]	1 prover	Yes	No
Wang et al. [74]	2 parties	No	Yes
VDORAM (ours)	**≥ 1 provers**	**Yes**	**Yes**

VDORAM achieves V+DO



VDORAM is the *slowest* RAM

(You are allowed to laugh here)

Conclusion

- Some applications need verifiability and confidentially among multi users
- MPC made for ZKP constraints
- CompatCircuit: combine MPC with ZKP. Unified representation.
- VDORAM: a vRAM (zkVM) and a DORAM (RAM-MPC)
- Future work:
  - Performance optimization in time cost
  - Performance optimization in networking
  - Proof compression
  - Real-world applications based on CompatCircuit / VDORAM

Thank you!

- Questions and Comments?

Huayi Qi

Shandong University

Tsinghua University

qi@huayi.email