# Gage MPC: Bypassing Residual Function Leakage for Non-Interactive MPC

**Ghada Almashaqbeh**[1] Fabrice Benhamouda[2] Seungwook Han[3]
Daniel Jaroslawicz[3] Tal Malkin[3] Alex Nicita[3] Tal Rabin[4,2]
Abhishek Shah[3] Eran Tromer[3,5]

[1]UConn, [2]Algorand, [3]Columbia, [4]UPenn, [5]Tel-Aviv University

# Gage MPC

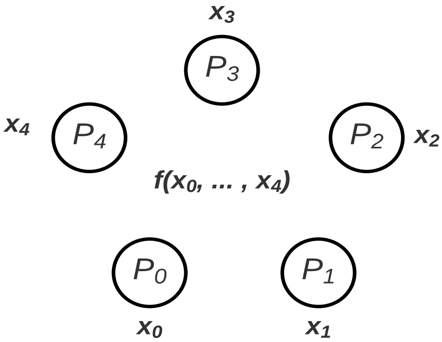
- A new model combining MPC and blockchains/smart contracts
- Circumvent lower bounds in NIMPC

# Gage MPC

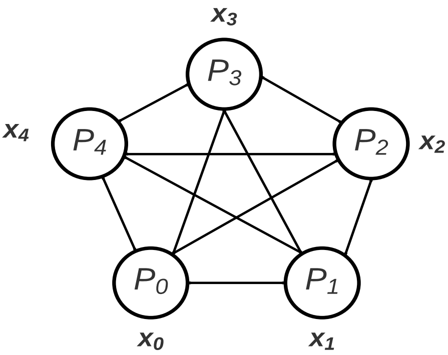
- A new model combining MPC and blockchains/smart contracts
- Circumvent lower bounds in NIMPC

MPC? NIMPC? MPC and blockchains?

# Multiparty Computation (MPC)

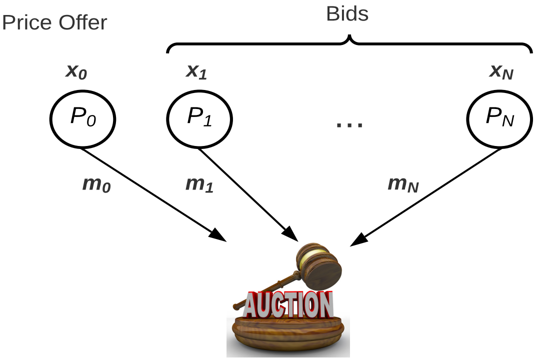


# Multiparty Computation (MPC)




Interactive, available!

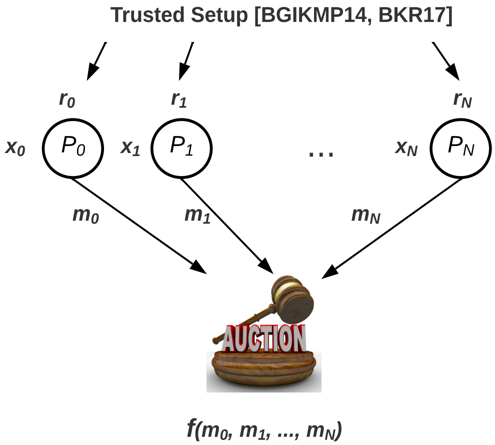
# Non-interactive MPC (NIMPC)




$f(x_0, x_1, ..., x_N)$

**The winner is ...**

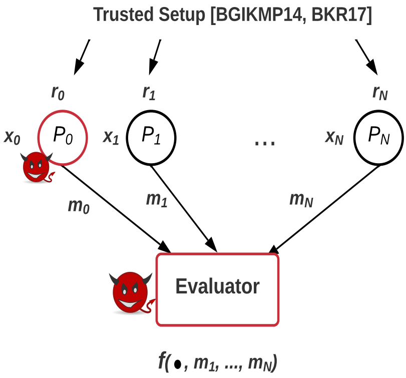
# Non-interactive MPC (NIMPC)




**The winner is ...**

# NIMPC Lower Bounds

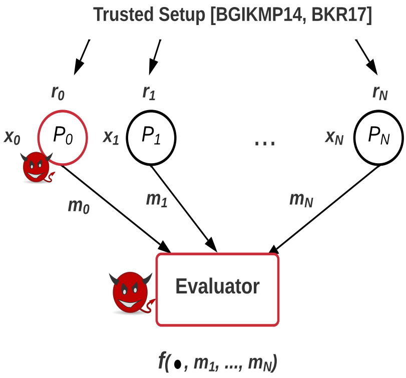
- Leakage of the *Residual Function* is inherent
  Evaluator and say $P_0$ can compute $f(\bullet, m_1, \ldots, m_N)$



# NIMPC Lower Bounds

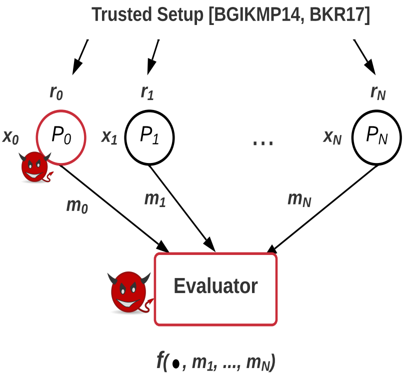
- Leakage of the *Residual Function* is inherent
  Evaluator and say $P_0$ can compute $f(\bullet, m_1, \ldots, m_N)$

- Setup assumptions: pre-shared randomness and a dedicated party to do the computation



# NIMPC Lower Bounds

- Leakage of the *Residual Function* is inherent
  Evaluator and say $P_0$ can compute $f(\bullet, m_1, \ldots, m_N)$
- Setup assumptions: pre-shared randomness and a dedicated party to do the computation




*Avoid such limitations??!*

# MPC and Blockchains

- **Gen I**. A blockchain implements a broadcast channel

# MPC and Blockchains

- **Gen I**. A blockchain implements a broadcast channel
- **Gen II**. Payments are incorporated into MPC

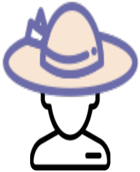
# MPC and Blockchains

- **Gen I**. A blockchain implements a broadcast channel
- **Gen II**. Payments are incorporated into MPC
- **Gen III**. *This work; Gage MPC!*
  - Smart contracts and miners are active participants in MPC
  - Circumvent the residual function leakage in NIMPC

# Gen II: Circumvent Fairness Lower Bound

$x_1$

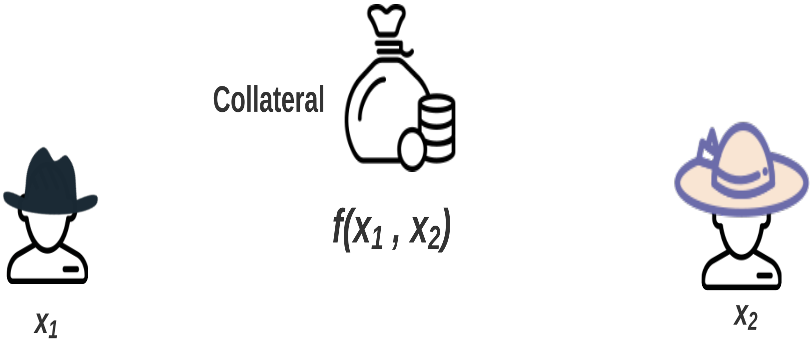
$f(x_1, x_2)$



$x_2$

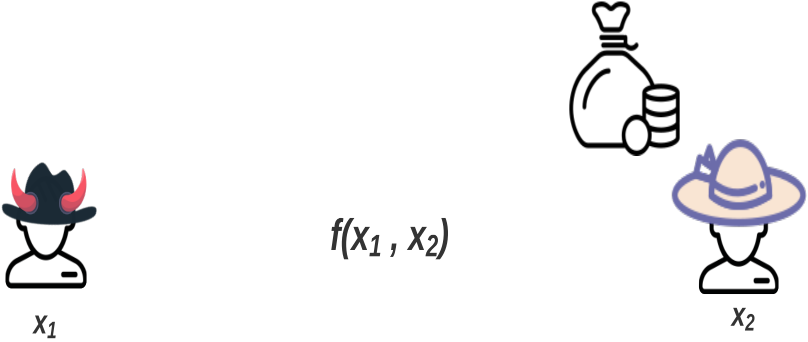
Fairness: either all get the output or none

# Gen II: Circumvent Fairness Lower Bound




Collateral is large enough to incentivize Bob to complete the computation

# Gen II: Circumvent Fairness Lower Bound




Bob may forgo his collateral → Not a complete fairness!

# Gen III: Gage MPC

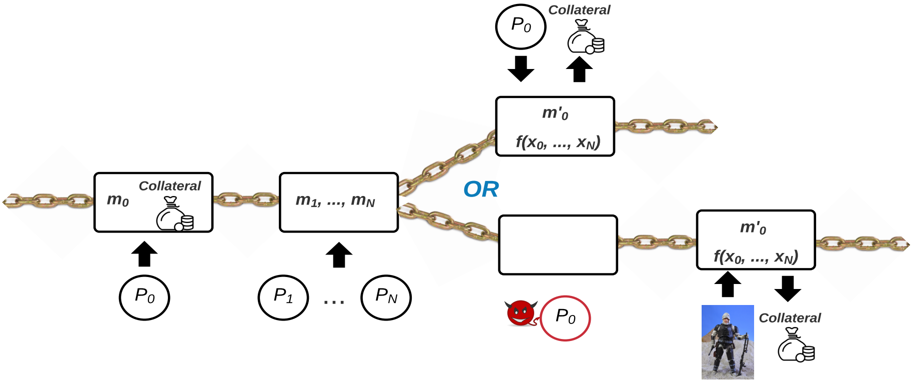
$x_1$

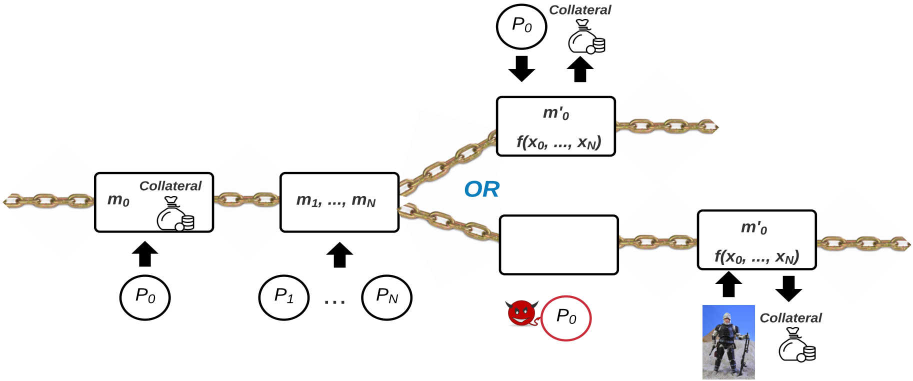
$$f(x_1 , x_2)$$

$x_2$

Complete fairness!

# Gage MPC


$m_0$
Collateral
$P_0$
$m_1, ..., m_N$
$P_1$
...
$P_N$
$P_0$
Collateral
$m'_0$
$f(x_0, ..., x_N)$
OR
$P_0$
$m'_0$
$f(x_0, ..., x_N)$
Collateral




A monetary assumption. *An honest party can put a collateral of value much higher than what an adversary can expend on computation.*

# On Circumventing NIMPC Lower Bounds

- Eliminate the leakage of the residual function
  - Re-valuating $f$ on a different set of inputs is very costly (same amount of the collateral)

# On Circumventing NIMPC Lower Bounds

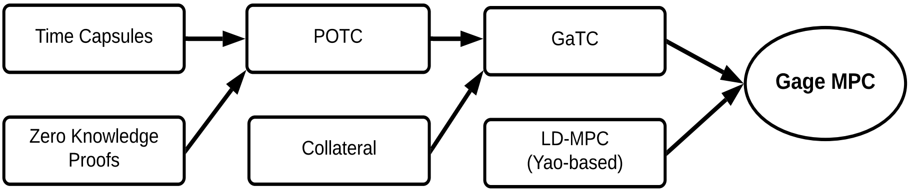
- Eliminate the leakage of the residual function
  - Re-valuating $f$ on a different set of inputs is very costly (same amount of the collateral)
- Eliminate setup assumptions.
  - No PKI or pre-shared randomness
  - No need for a dedicated online evaluator

# On Circumventing NIMPC Lower Bounds

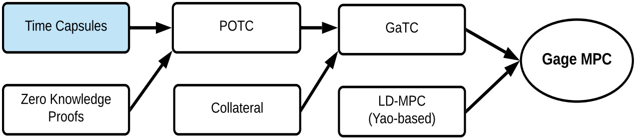
- Eliminate the leakage of the residual function
  - Re-valuating $f$ on a different set of inputs is very costly (same amount of the collateral)
- Eliminate setup assumptions.
  - No PKI or pre-shared randomness
  - No need for a dedicated online evaluator

Gage MPC guarantees *short term* security!

# Gage MPC: Our Construction



# Time Capsules




Simply commit to a value that can be force-opened after expending a pre-specified amount of computation

E.g., $h(s)$ where $s \leftarrow \{0, 1\}^{\lambda^*}$

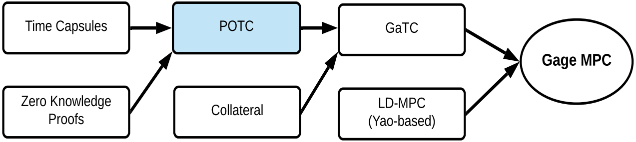
# Time Capsules Definition

- Commit: $(c, d) \leftarrow TC.Commit(1^\lambda, 1^{\lambda^*}, m)$
  - $\lambda$ is the regular security parameter
  - $\lambda^*$ sets the complexity for force open
- Decommit Verify: $TC.DVrfy(1^\lambda, c, d, m)$ outputs 1 if $d$ is a valid opening with respect to $c$ and $m$
- Forced Open: $(m, d) = TC.ForceOpen(c)$ brute-forces the opening of $c$

# Time Capsules Properties

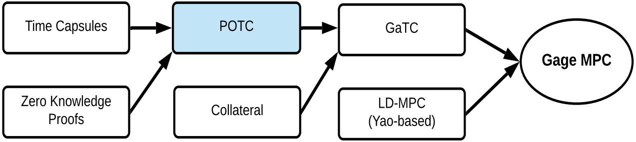
- Correctness
- Binding
- Hiding:
  - Related computation required to force open
  - For any adversary with computation less than $2^{\lambda^*}$, the capsule should remain hiding

# Proof of Time Capsules (POTC)




How about front running?

# Proof of Time Capsules (POTC)




How about front running?

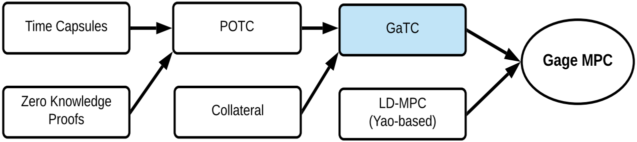
POTC:

- Instead of announcing the decommitment itself (i.e., $d$ and $m$), prove in zero knowledge that $d$ has been found and announce $m$
- Connect the opening to the miner's wallet or public key via a tag
- Proof Verify: $TC.PVrfy(1^\lambda, c, m, \pi, tag)$ outputs 1 if $\pi$ is correct with respect to $c, m$ and $tag$

# Is POTC Enough?

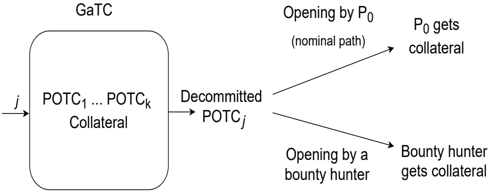
- How to reward for force-open?
- How to choose $m$ while the other party's input is not known yet?

# Gage Time Capsules (GaTC)




Bundle several POTCs together

A smart contract will manage the collateral (force-open award)

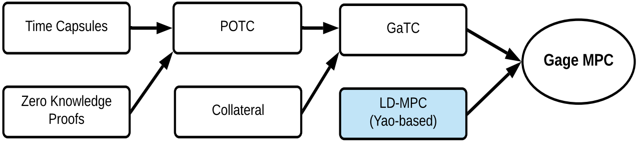


# The Computation?

POTC takes care of hiding the input labels

How to perform the computation using these labels?

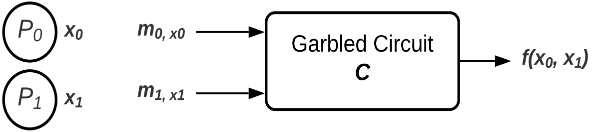
# Label Driven MPC (LD-MPC)




A generalization of Garbled Circuits that is robust to the exposure of additional labels.

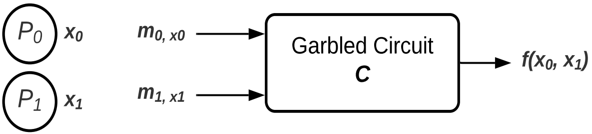
# Label Driven MPC (LD-MPC)

Conventional Yao; Exposure of any additional label compromises privacy

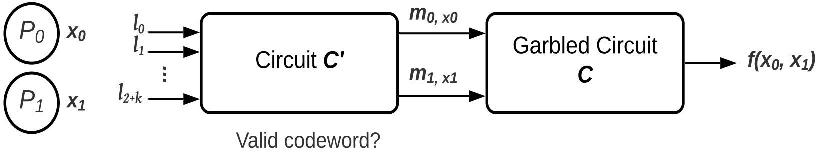


# Label Driven MPC (LD-MPC)

Conventional Yao; Exposure of any additional label compromises privacy




LD-MPC = Linear Error Correcting Codes + Yao



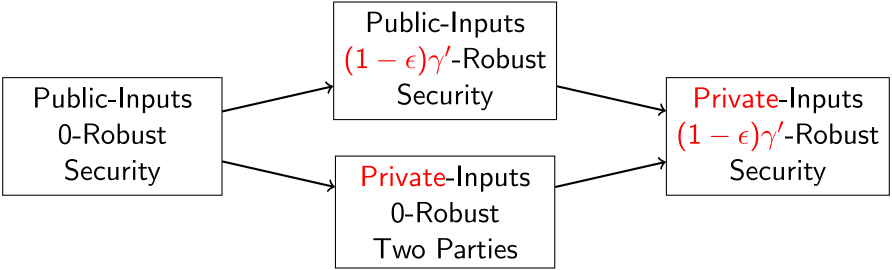
# Main Result — Gage MPC

Combines LD-MPC with GaTC

Simplest case; Only $P_0$'s input is private

- $P_0$ prepares a garbled circuit, GaTCs for input labels for $P_1, \ldots, P_N$, and a controller smart contract
- $P_1, \ldots, P_N$ submit their inputs
- Either $P_0$ will come back and open the corresponding labels, or bounty hunters will do
- Smart contract (aka blockchain miners) evaluate the circuit over the labels and record the output

# Gage MPC — Transformations




The private input versions support only two party computation

# Conclusion

## Main Result — Gage MPC

NIMPC for $f$ leaking $R$ and requiring $TS$ $\rightarrow$ NIMPC with no $R$ and $TS$

Gen III of MPC + blockchain

## Side Result

Several new primitives (POTC, GaTC, and LD-MPC) that could be of independent interest

Thank you!

Questions?