GAuV: A Graph-Based Automated Verification Framework for Perfect Semi-Honest Security of Multiparty Computation Protocols

Xingyu Xie*, Yifei Li*, Wei Zhang, Tuowei Wang, Shizhen Xu, Jun Zhu, Yifan Song

Tsinghua University

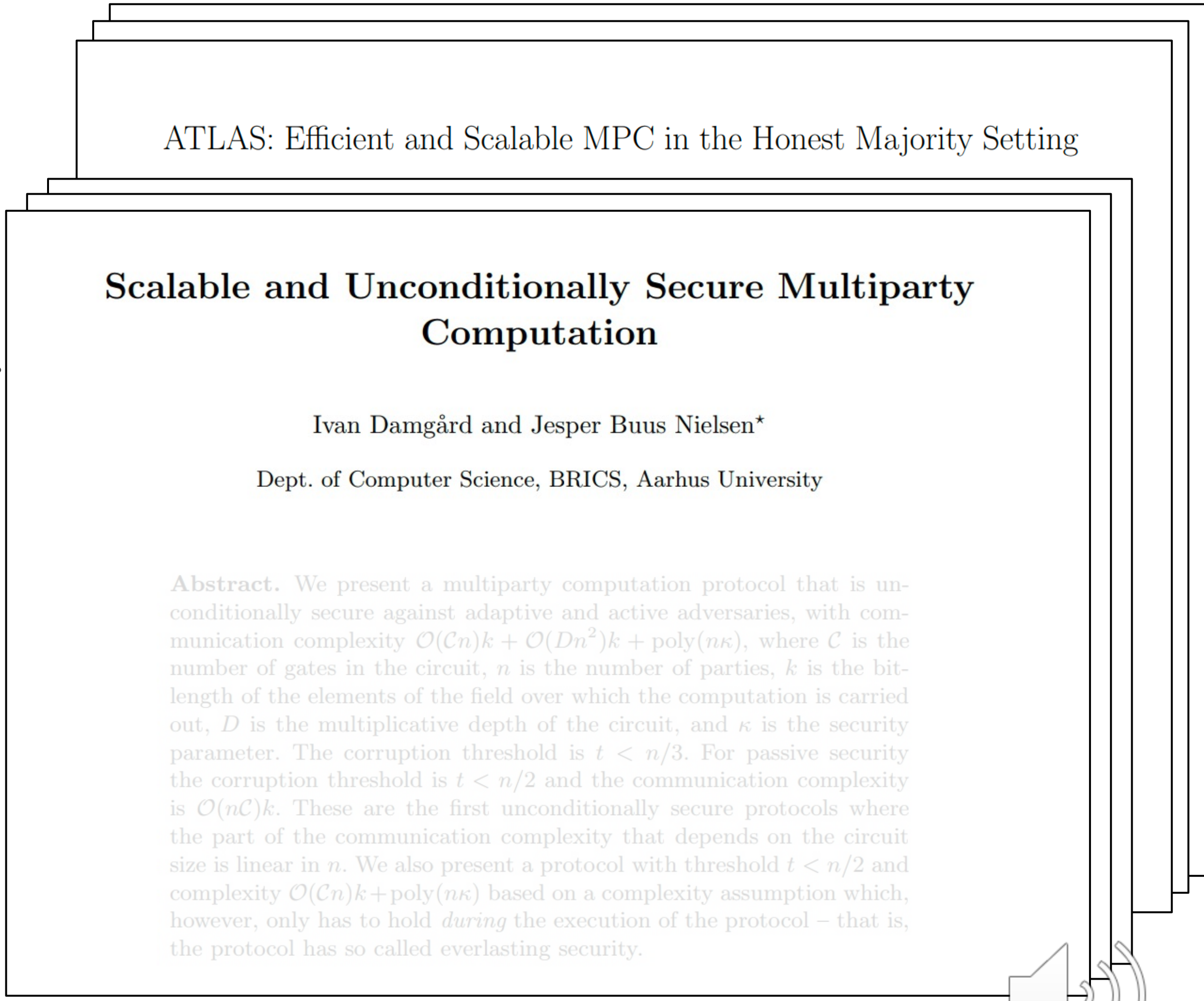
RealAI

Background: Multiparty Computation Needs Automated Verification

- The simulation-based security proof for multiparty computation is tricky, and not easy to understand.

- If you are tired of reading and writing tedious proofs…
- If you are afraid of unnoticed mistakes in the proof…

we're here to help!



ATLAS: Efficient and Scalable MPC in the Honest Majority Setting

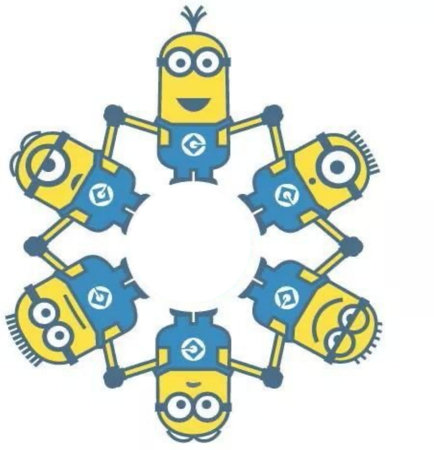
Scalable and Unconditionally Secure Multiparty Computation

Ivan Damgård and Jesper Buus Nielsen*

Dept. of Computer Science, BRICS, Aarhus University

Abstract. We present a multiparty computation protocol that is unconditionally secure against adaptive and active adversaries, with communication complexity $\mathcal{O}(\mathcal{C}n)k + \mathcal{O}(Dn^2)k + \text{poly}(n\kappa)$, where $\mathcal{C}$ is the number of gates in the circuit, n is the number of parties, k is the bit-length of the elements of the field over which the computation is carried out, D is the multiplicative depth of the circuit, and κ is the security parameter. The corruption threshold is $t < n/3$. For passive security the corruption threshold is $t < n/2$ and the communication complexity is $\mathcal{O}(n\mathcal{C})k$. These are the first unconditionally secure protocols where the part of the communication complexity that depends on the circuit size is linear in n. We also present a protocol with threshold $t < n/2$ and complexity $\mathcal{O}(\mathcal{C}n)k+\text{poly}(n\kappa)$ based on a complexity assumption which, however, only has to hold *during* the execution of the protocol – that is, the protocol has so called everlasting security.

Tool Overview



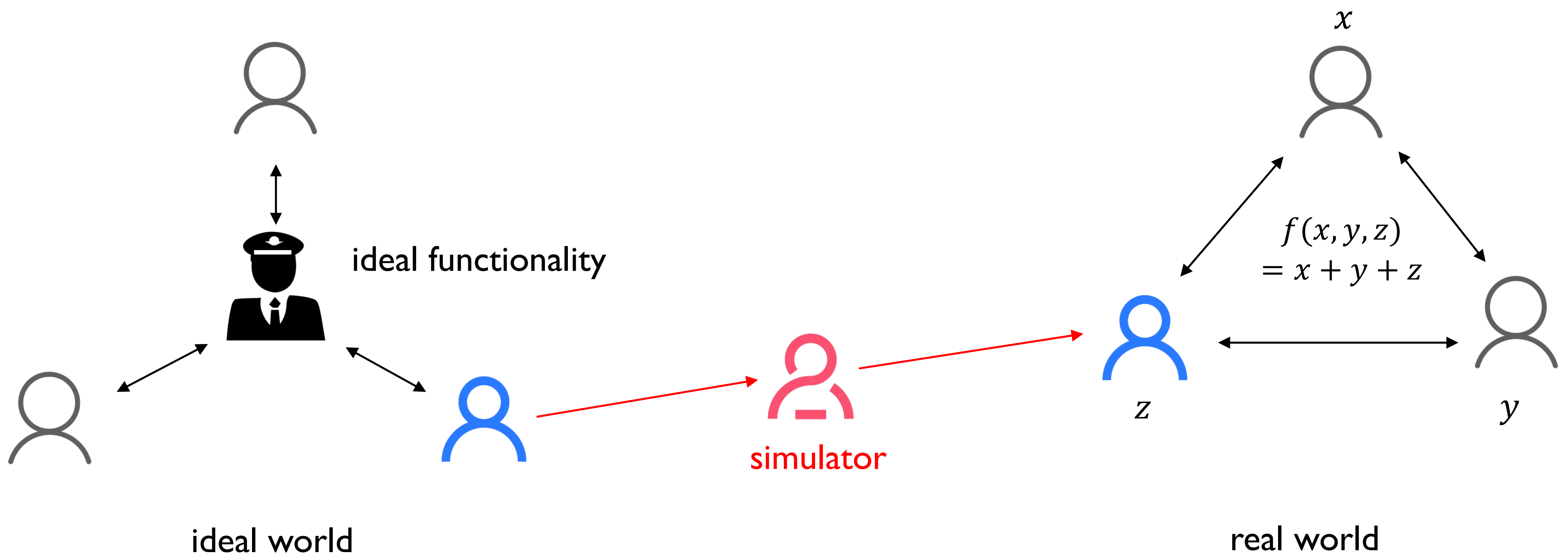
multiparty computation protocol

GAuV

✓ (secure)

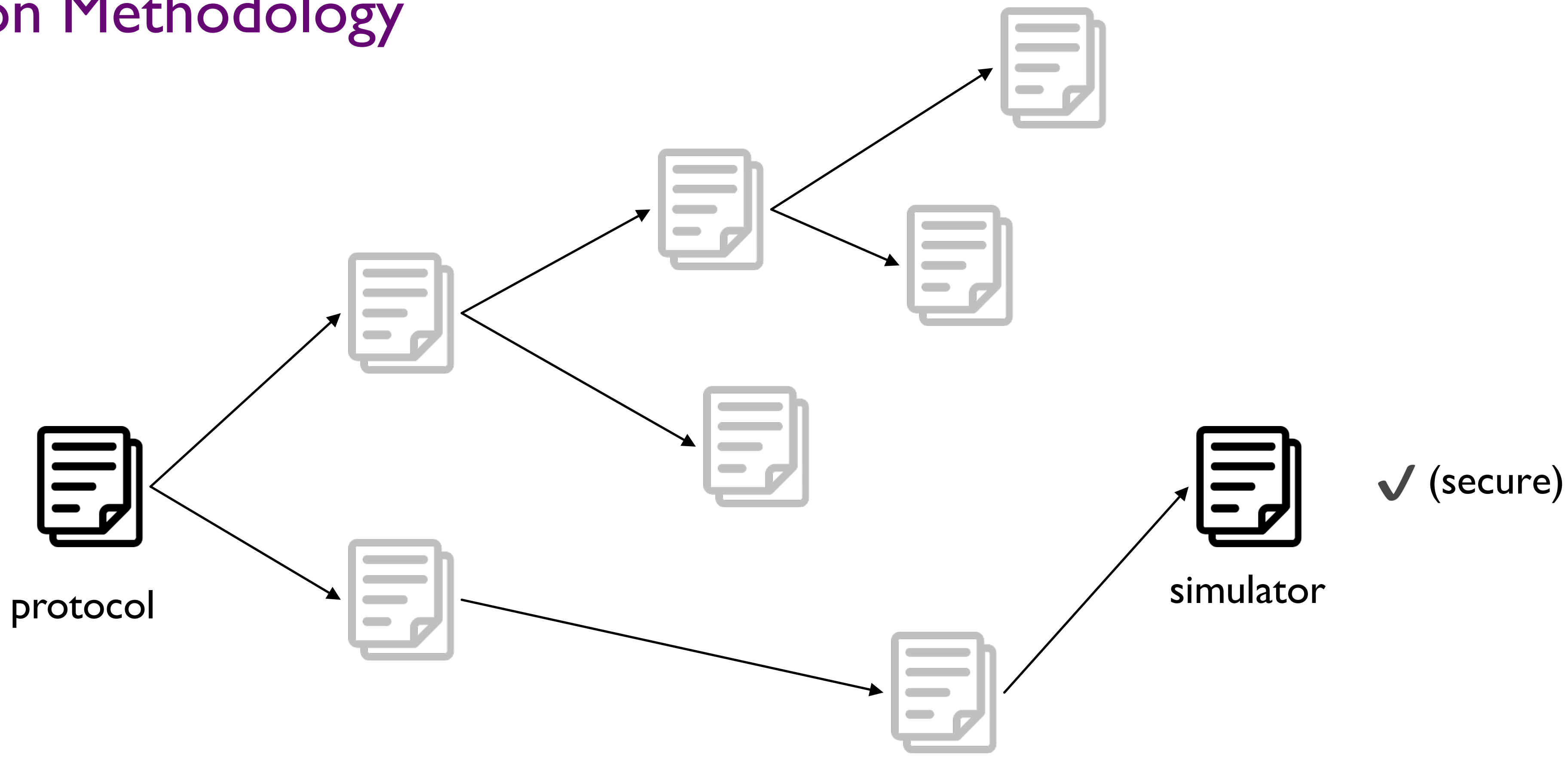
? (unknown)

Preliminary: Simulation-based Security for Multiparty Computation



Security Definition: there exists an algorithm that simulates the views of corrupted parties only from corrupted parties' inputs and outputs.

Verification Methodology



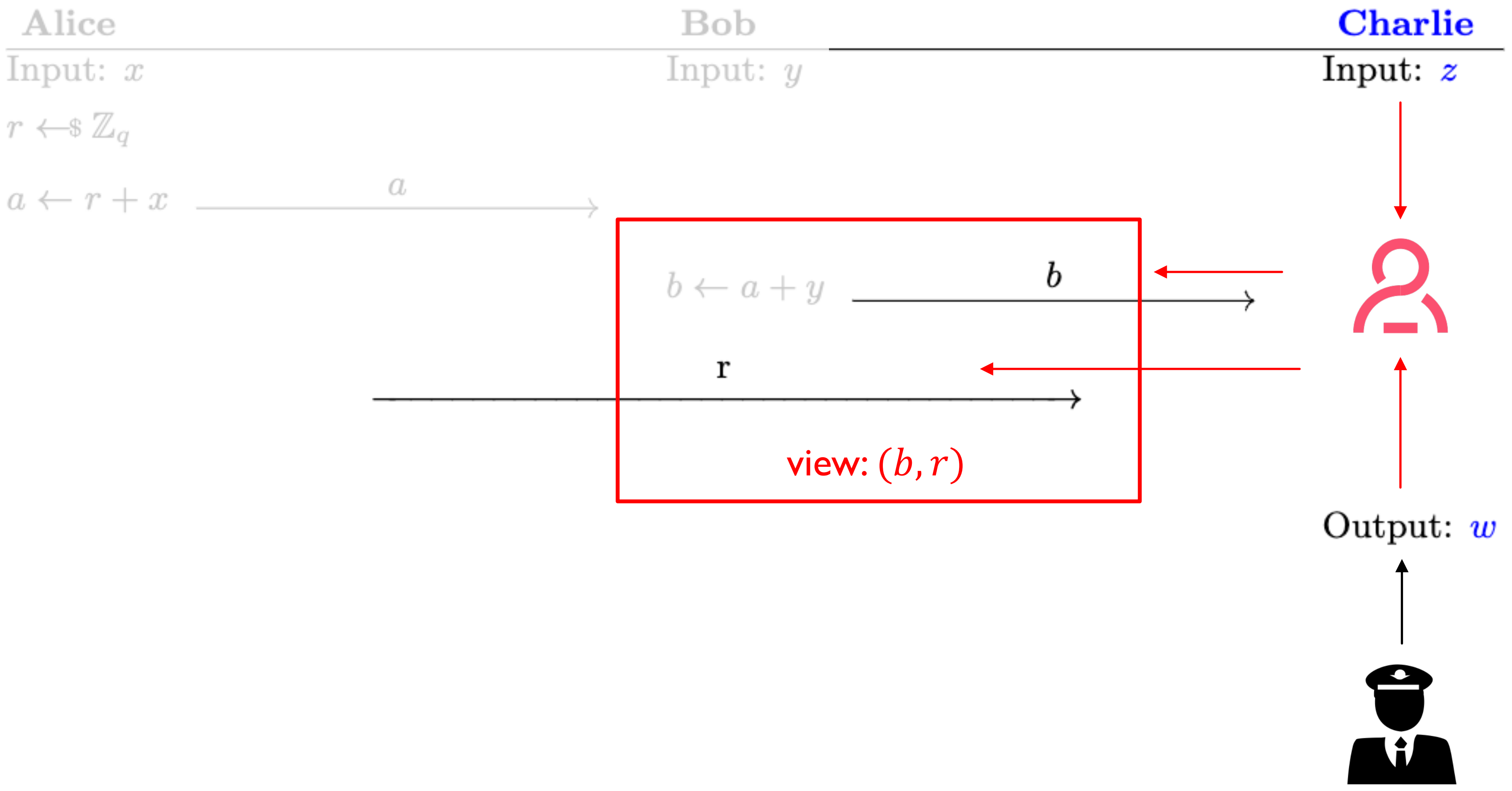
Methodology: try to transform the protocol into a simulator

3-Party Addition Protocol

Alice		Bob		Charlie
Input: x		Input: y		Input: z
$r \leftarrow\$ \mathbb{Z}_q$				
$a \leftarrow r + x$	$\xrightarrow{a}$			
		$b \leftarrow a + y$	$\xrightarrow{b}$	$c \leftarrow b + z$
	$\xrightarrow{\text{r}}$ (Alice to Charlie)			
				$w \leftarrow c - r$
				Output: w

Ideal functionality: $f(x, y, z) = (\lambda, \lambda, x + y + z)$

Simulator of Charlie



Technical Goal: to Transform the Protocol to the Simulator

G_1 (Protocol)

$r \leftarrow\$ \mathbb{Z}_q$
$a \leftarrow r + x$
$b \leftarrow a + y$
$c \leftarrow b + z$
$w \leftarrow c - r$
return (b, r)

Simulator

Input: z, w

$\ldots$

return (b, r)

Representation: Data-flow Graph

G_1 (Protocol)

$r \leftarrow_{\$} \mathbb{Z}_q$

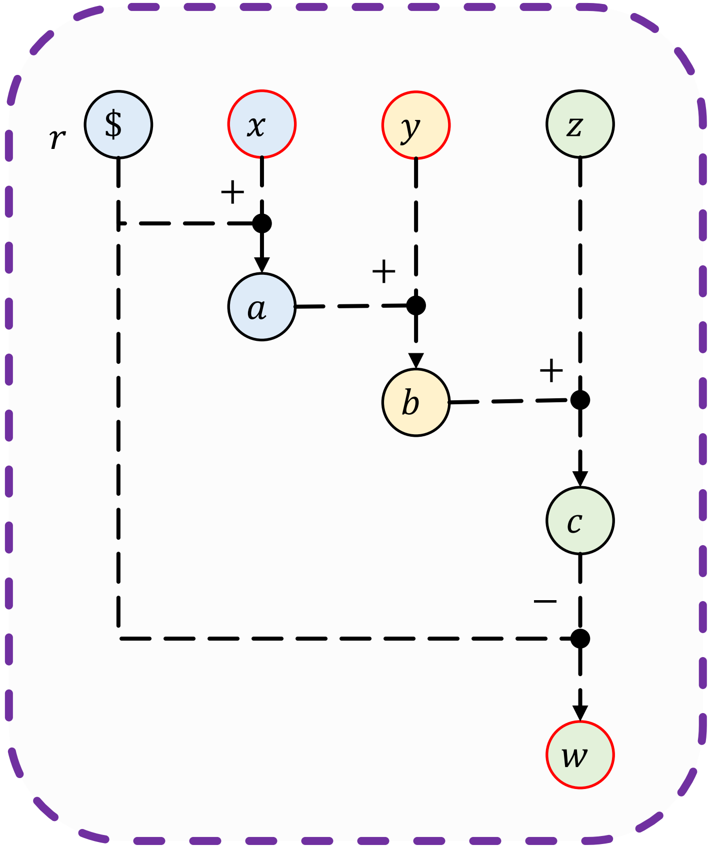
$a \leftarrow r + x$

$b \leftarrow a + y$

$c \leftarrow b + z$

$w \leftarrow c - r$

$\mathbf{return}\ (b, r)$



Simulator

Input: z, w

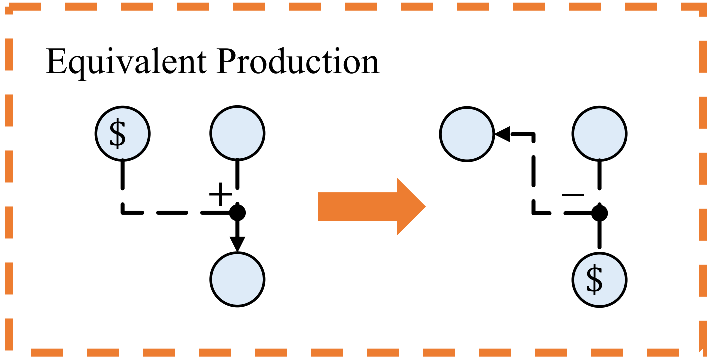
$\ldots$

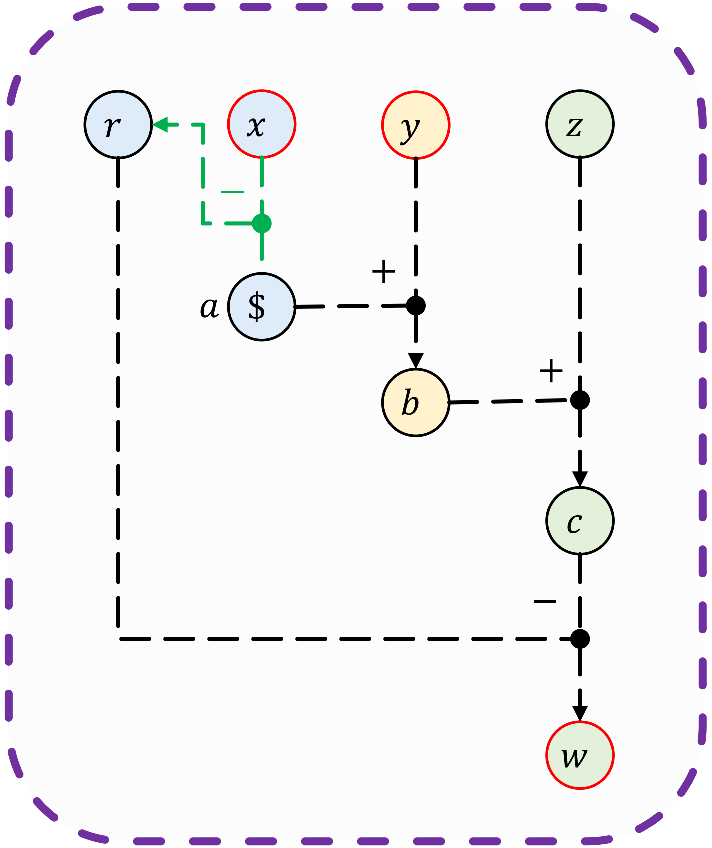
$\mathbf{return}\ (b, r)$

Technical Goal: to remove the dependency on the honest parties' inputs and outputs, while preserving the distribution of the corrupted parties' views during the transformation.

Proof Step 1

G_1 (Protocol)	G_2
$r \leftarrow\$ \mathbb{Z}_q$	$a \leftarrow\$ \mathbb{Z}_q$
$a \leftarrow r + x$	$r \leftarrow a - x$
$b \leftarrow a + y$	$b \leftarrow a + y$
$c \leftarrow b + z$	$c \leftarrow b + z$
$w \leftarrow c - r$	$w \leftarrow c - r$
return (b, r)	**return** (b, r)





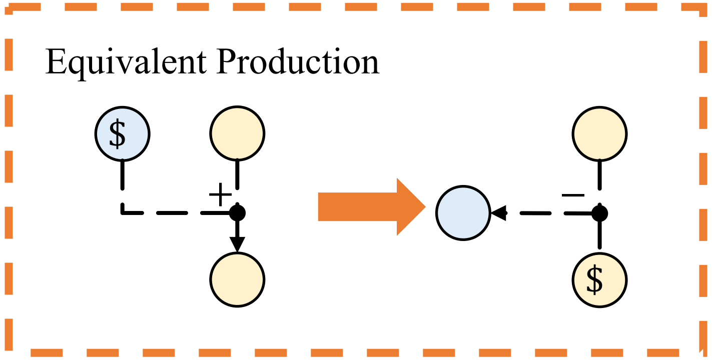
Simulator

Input: z, w

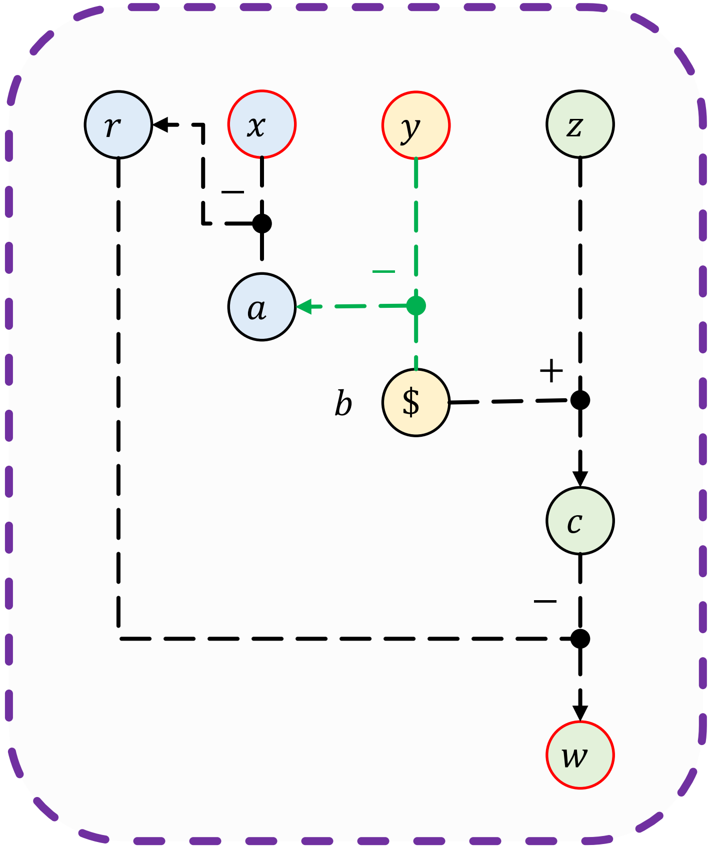
...

return (b, r)

Proof Step 2



G_1 (Protocol)	G_2	G_3
$r \leftarrow\$ \mathbb{Z}_q$	$a \leftarrow\$ \mathbb{Z}_q$	$b \leftarrow\$ \mathbb{Z}_q$
$a \leftarrow r + x$	$r \leftarrow a - x$	$a \leftarrow b - y$
$b \leftarrow a + y$	$b \leftarrow a + y$	$r \leftarrow a - x$
$c \leftarrow b + z$	$c \leftarrow b + z$	$c \leftarrow b + z$
$w \leftarrow c - r$	$w \leftarrow c - r$	$w \leftarrow c - r$
return (b, r)	**return** (b, r)	**return** (b, r)



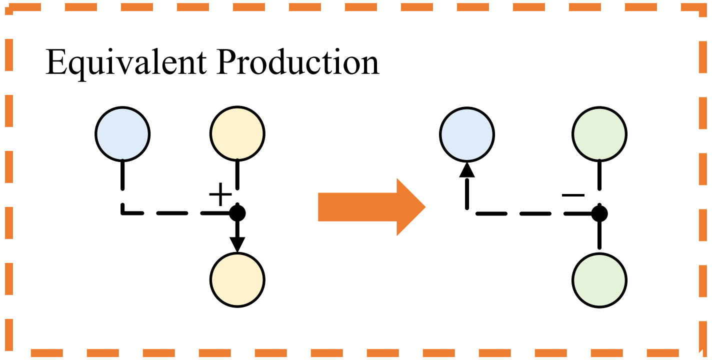
Simulator

Input: z, w

$\dots$

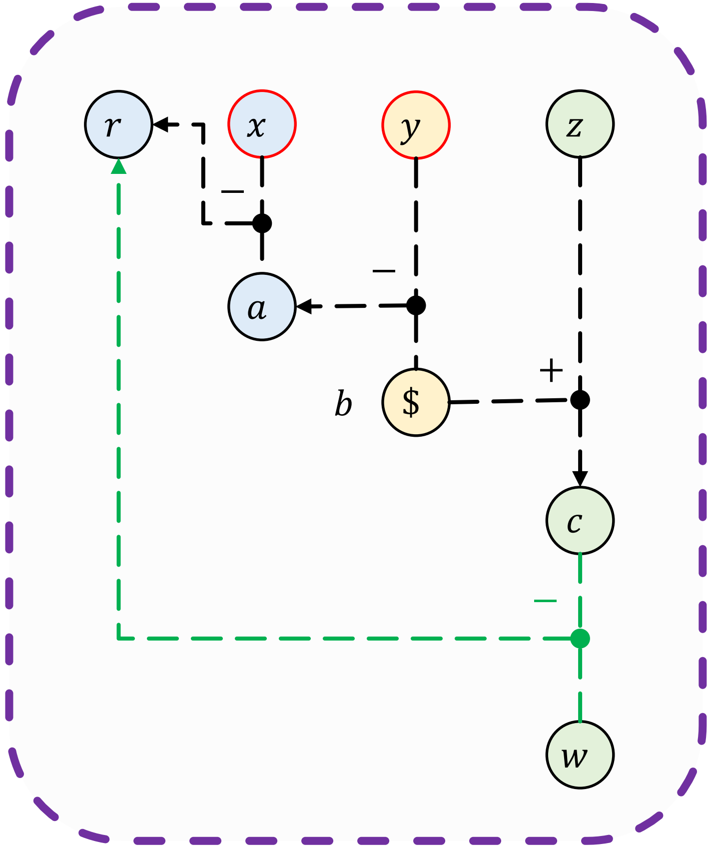
return (b, r)

Proof Step 3



G_1 (Protocol)	G_2	G_3	G_4
$r \leftarrow\$ \mathbb{Z}_q$	$a \leftarrow\$ \mathbb{Z}_q$	$b \leftarrow\$ \mathbb{Z}_q$	$b \leftarrow\$ \mathbb{Z}_q$
$a \leftarrow r + x$	$r \leftarrow a - x$	$a \leftarrow b - y$	$a \leftarrow b - y$
$b \leftarrow a + y$	$b \leftarrow a + y$	$r \leftarrow a - x$	$c \leftarrow b + z$
$c \leftarrow b + z$	$c \leftarrow b + z$	$c \leftarrow b + z$	$r \leftarrow a - x$
$w \leftarrow c - r$	$w \leftarrow c - r$	$w \leftarrow c - r$	$r \leftarrow c - w$
return (b, r)	**return** (b, r)	**return** (b, r)	**return** (b, r)
			$w \leftarrow x + y + z$

We can use the ideal functionality's output, if the protocol is correct, and the correctness is preserved during transformation.



Simulator

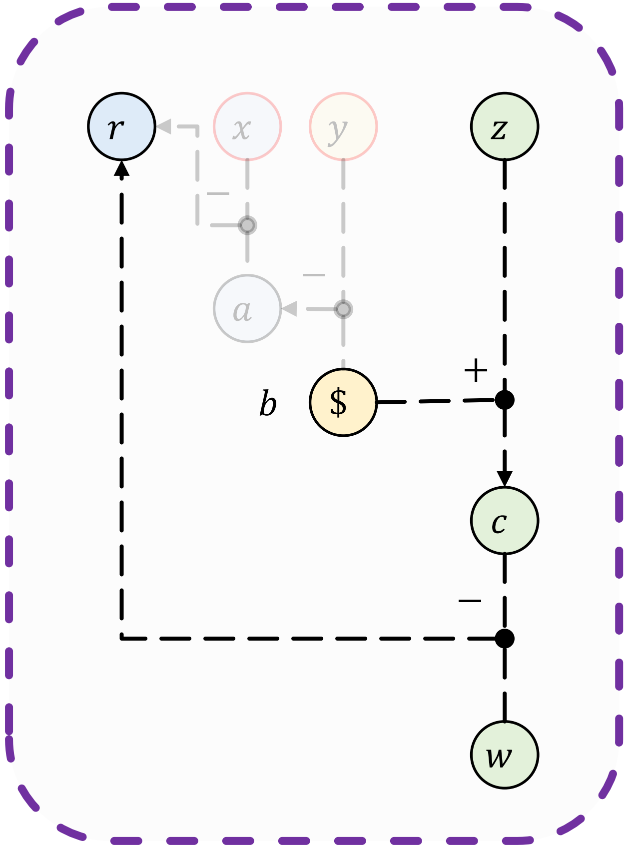
Input: z, w

...

return (b, r)

Proof Step 4

G_1 (Protocol)	G_2	G_3	G_4	G_5 (Simulator)
$r \leftarrow\$ \mathbb{Z}_q$	$a \leftarrow\$ \mathbb{Z}_q$	$b \leftarrow\$ \mathbb{Z}_q$	$b \leftarrow\$ \mathbb{Z}_q$	$b \leftarrow\$ \mathbb{Z}_q$
$a \leftarrow r + x$	$r \leftarrow a - x$	$a \leftarrow b - y$	$a \leftarrow b - y$	$c \leftarrow b + z$
$b \leftarrow a + y$	$b \leftarrow a + y$	$r \leftarrow a - x$	$c \leftarrow b + z$	$r \leftarrow c - w$
$c \leftarrow b + z$	$c \leftarrow b + z$	$c \leftarrow b + z$	$r \leftarrow a - x$	**return** (b, r)
$w \leftarrow c - r$	$w \leftarrow c - r$	$w \leftarrow c - r$	$r \leftarrow c - w$	$w \leftarrow x + y + z$
return (b, r)	**return** (b, r)	**return** (b, r)	**return** (b, r)	
			$w \leftarrow x + y + z$	

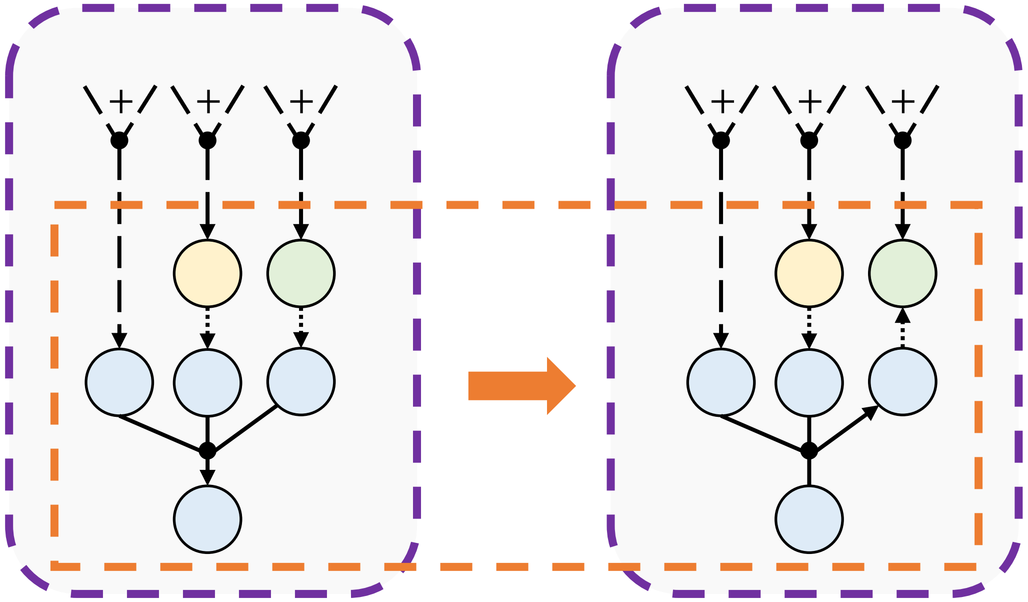


“Good” transformations

Definition (equivalent rewriting)

Given an equivalent production (L, R) and a structured-preserving mapping $f: L \rightarrow G$, transform G to H by substituting $f(L)$ with R, and check if

- both G and H are acyclic;
- the number of each type of the random nodes are preserved.



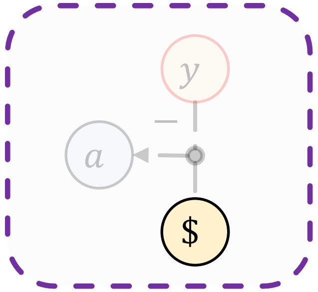
Definition (vintage transformation)

A *vintage transformation* from G to H is an injection from the nodes of H to G, for every assignment γ of the input and the output correct for G:

(i-iv) basic nodes and properties are preserved;
(v) the distribution of views is preserved;
(vi) H is correct w.r.t. γ.

Definition (tail node elimination)

Transform G to H by eliminating a node without out-edges.



Soundness

Definition (equivalent rewriting)

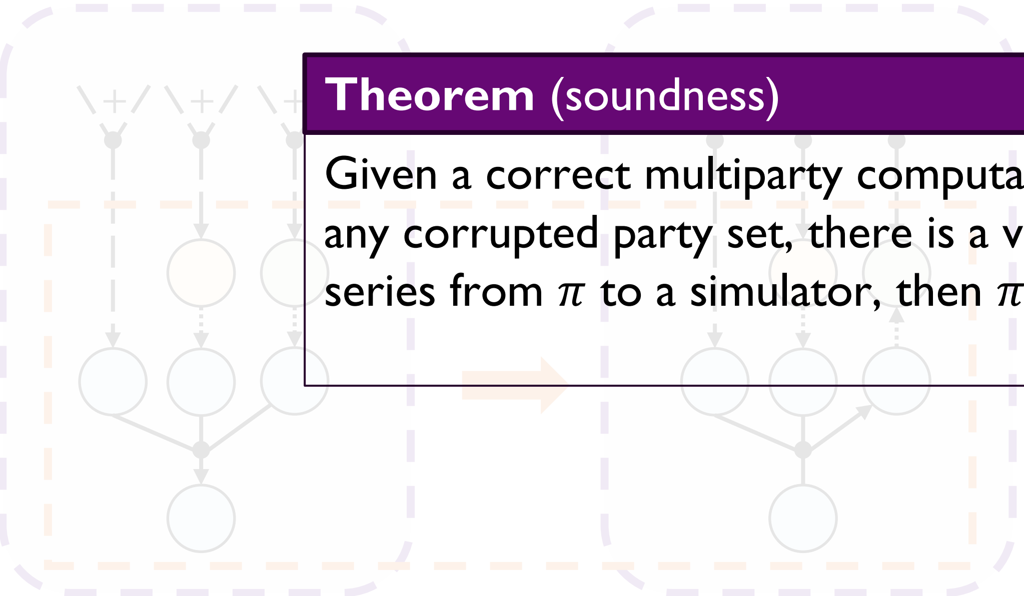
Given an equivalent production $p = (L, R)$ and a match morphism $f: L \to G$, transform G to H by substituting $f(L)$ with R, and check if

- both G and H are acyclic;
- the possibilities provided by random nodes are preserved.

Definition (vintage transformation)

The transformation from G to H is a vintage transformation, if there exists an injection h_v from the nodes of H to G, so that for every assignment γ on the input and output (s.t. G is correct w.r.t. γ):

(i-iv) basic nodes and properties are preserved;
(v) the distribution of views is preserved;
(vi) H is correct w.r.t. γ.



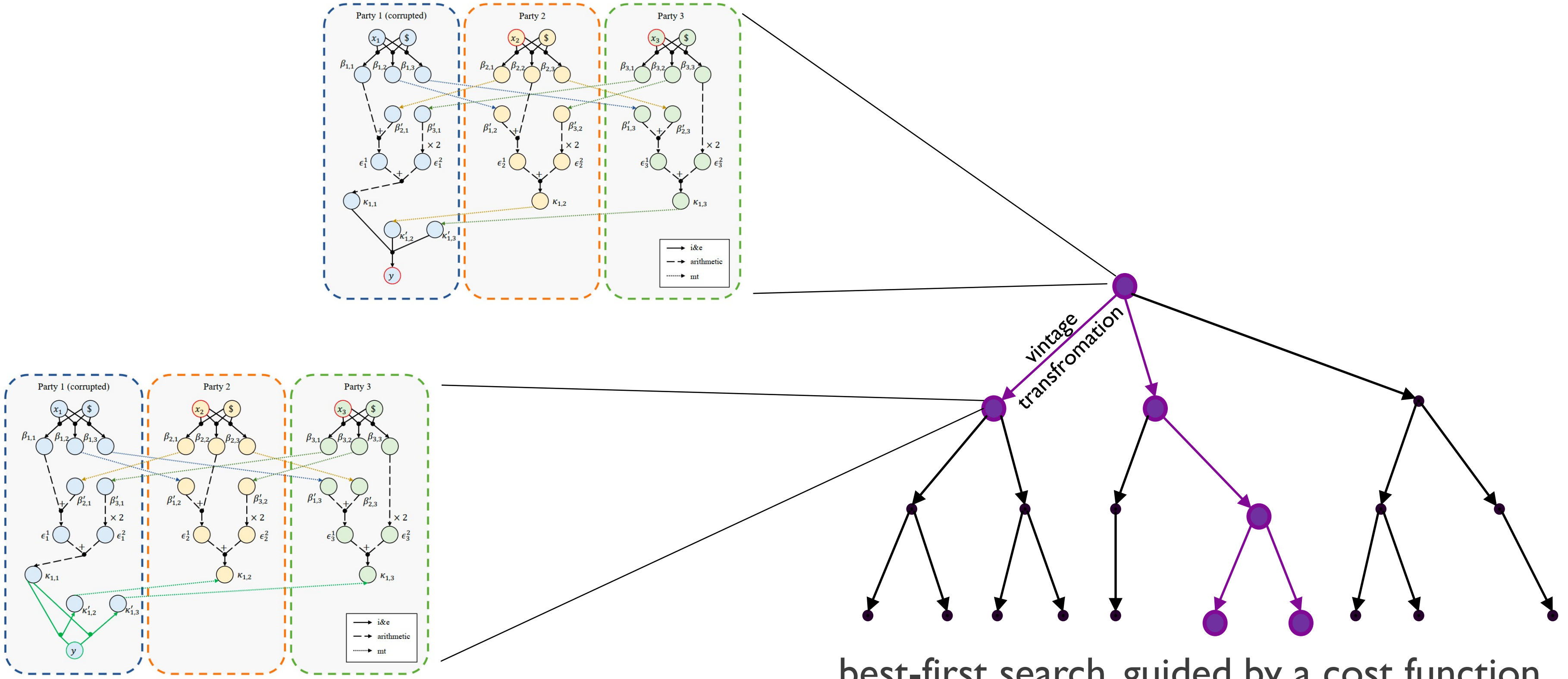
Theorem (soundness)

Given a correct multiparty computation protocol π, if, for any corrupted party set, there is a vintage transformation series from π to a simulator, then π is secure.

...il node elimination)

Transform G to H by eliminating a node without out-edges.

Algorithm



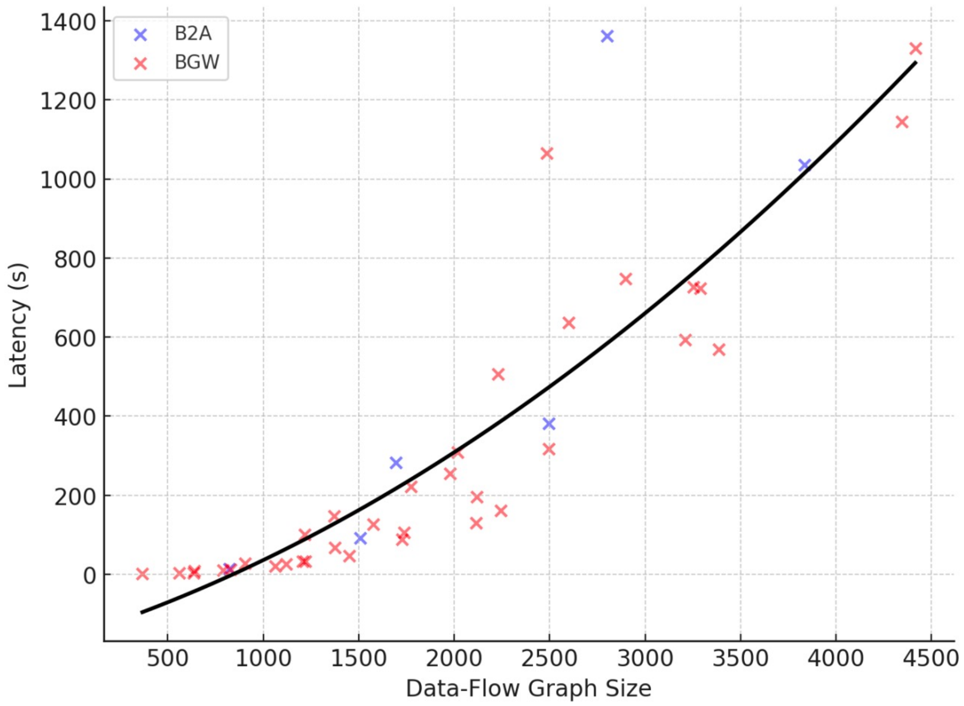
best-first search, guided by a cost function

Theorem (soundness)

Given a correct multiparty computation protocol π, if GAuV returns "secure" for any corrupted party set, then π is secure.

Evaluation

- Two cases from the literature
  - BGW (Ben-Or, Goldwasser and Wigderson) protocols
  - Binary-to-arithmetic secret sharing conversion



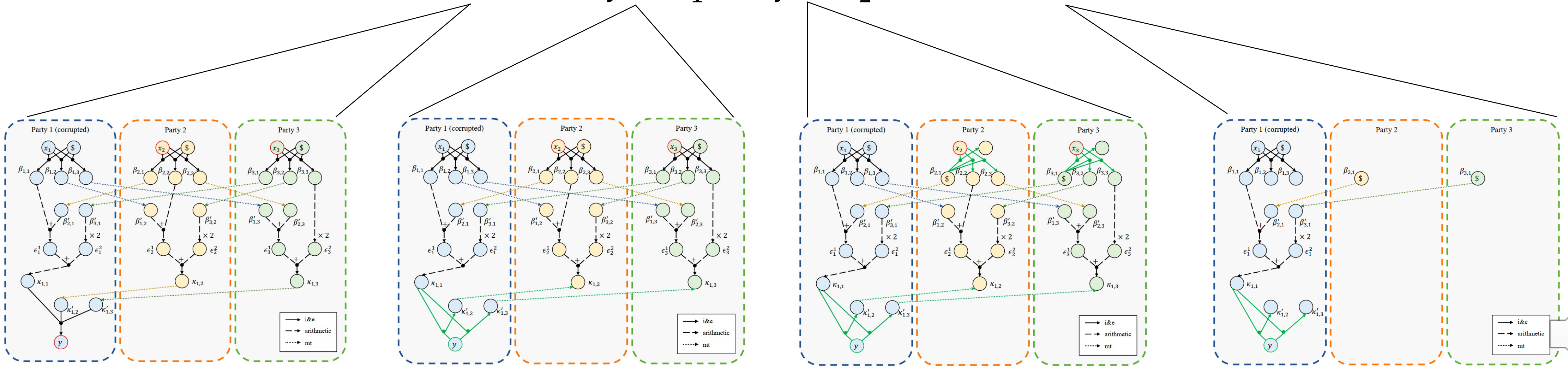
Theorem (completeness for BGW protocols)

Given a BGW protocol π, GAuV can prove its security for any corrupted party set in polynomial time, with suitable cost function.

On Generalization

- *Hybrid argument*: a common proof technique to show the indistinguishability between two distributions.
- For other more advanced perfectly secure protocols such as DN and ATLAS, the transformations between adjacent hybrids can be mechanized as equivalent rewriting or tail node elimination.
- Thus, GAuV can be used to prove the security of these protocols with sufficient amount of time.

$$Protocol = Hybrid_1 = Hybrid_2 = \cdots = Simulator$$



Summary

- An automated verification framework for proving the security of multiparty computation protocols
  - Security strength: perfect security
  - Threat model: semi-honest adversary
  - Condition: correctness of the protocol w.r.t. a deterministic ideal functionality
  - Trusted code base: self
- Open-sourced at https://github.com/leefige/gauv

Thanks!