Dumbo-MPC: Efficient Fully Asynchronous MPC with Optimal Resilience

Yuan Su[1], Yuan Lu[2], Jiliang Li[1], Yuyi Wang[3], Chengyi Dong[1], Qiang Tang[4]

[1]Xi'an Jiaotong University

[2]Institute of Software Chinese Academy of Sciences

[3]CRRC Zhuzhou Institute

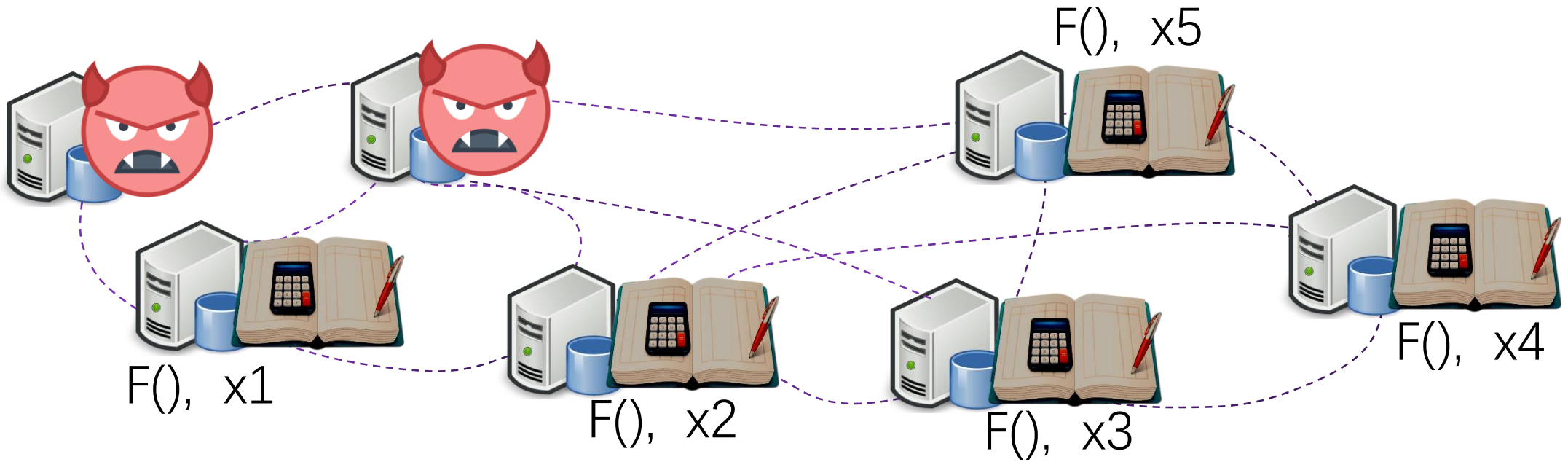
[4]The University of Sydney

contents

1 In Need of Practical Asynchronous MPC

Secure Multi-Party Computation

Collectively compute: **Y=F(x1, x2, x3, x4, x5)**



- Jointly and privately compute function
  - **Correctness (integrity):** correct evaluation result
  - **Privacy：** leaking nothing except the result

MPC-as-a-Service (MPCaaS)

MPCaaS with offline-online paradigm

Offline phase

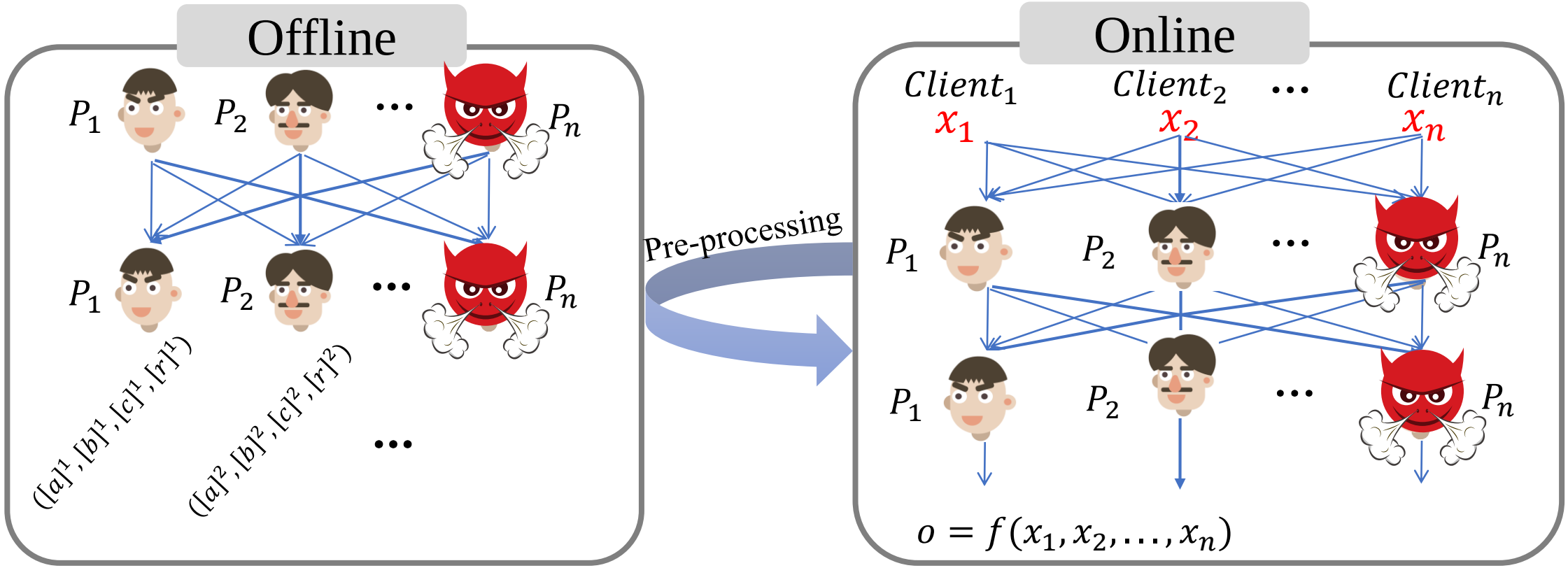
Beaver triples $([a], [b], [c])$ where $c = ab$

Random shares $[r]$

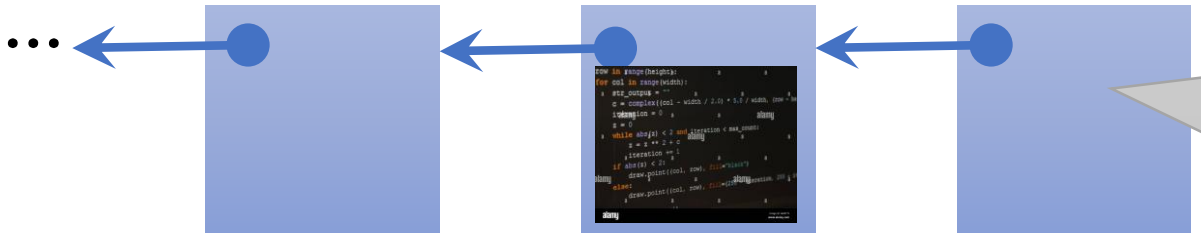
Online phase

Handle private inputs

Privately evaluate function f



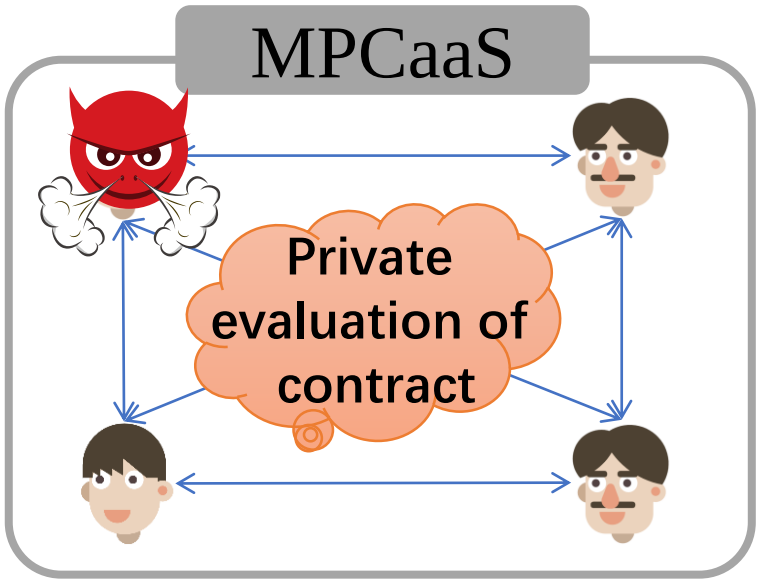
Call for Robustness (G.O.D.)



Transaction's cipher

TX Execution Result

In the application of
private smart contract,
we expect **G.O.D. !!!**
Otherwise, censorship !!!



Private Smart Contract

Enigma Project

6.5K Followers

Follow

Home About

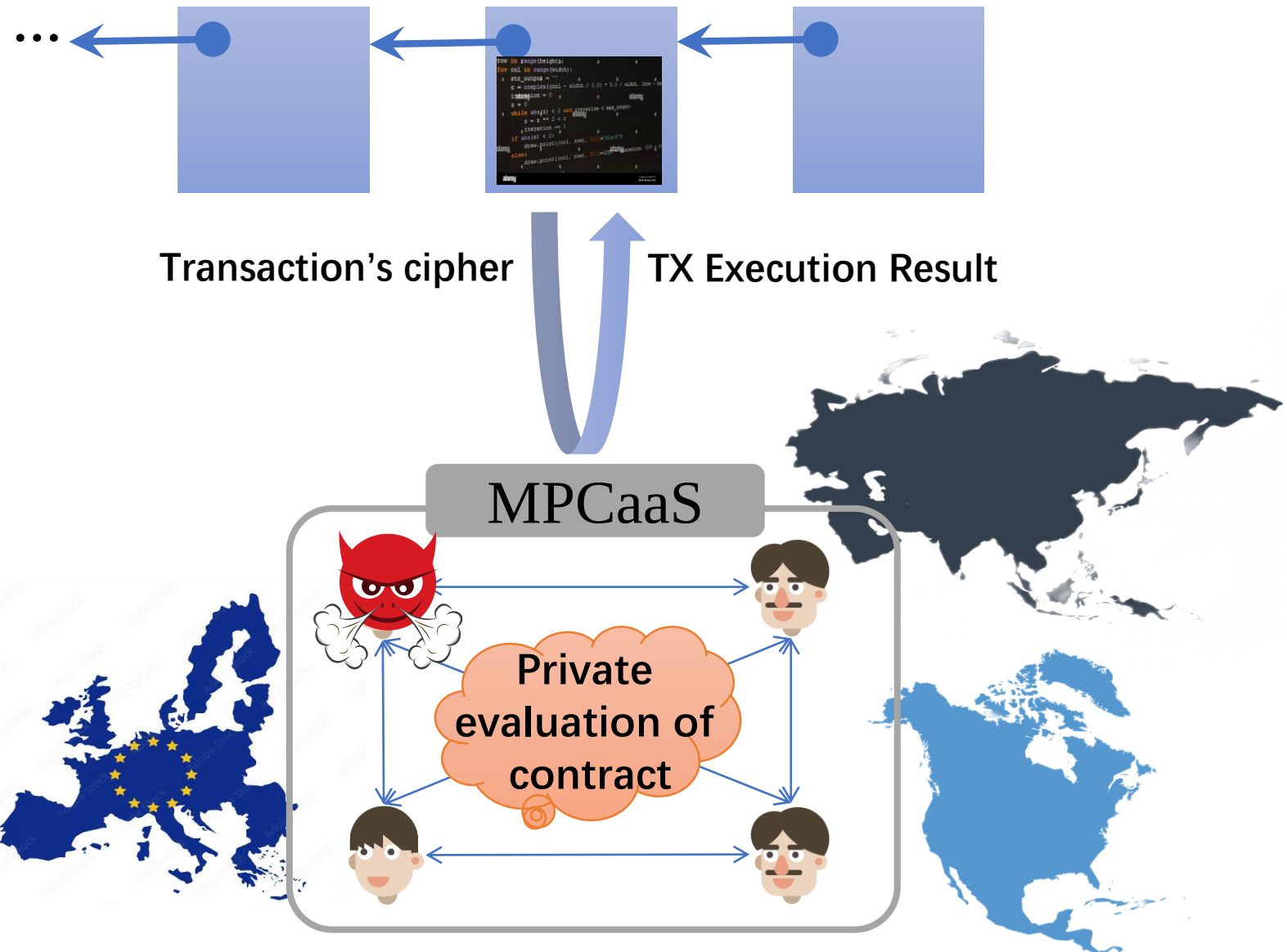
Enigma Project in Enigma

Introducing Secret Network

Our mainnet blockchain is now known as Secret Network, thanks t...

May 27, 2020 626 1

Call for Robustness (G.O.D.) in Asynchrony



Most Earlier MPCaaS:

In-house deployment in **Local Area Network** !!!

New Demand:

Global deployment over unreliable **Internet** !!!

- **Transmission interruption**
- **Fluctuating bandwidth**
- **Internet attacks**
- **Network misconfiguration**

Call for Robustness (G.O.D.) in Asynchrony

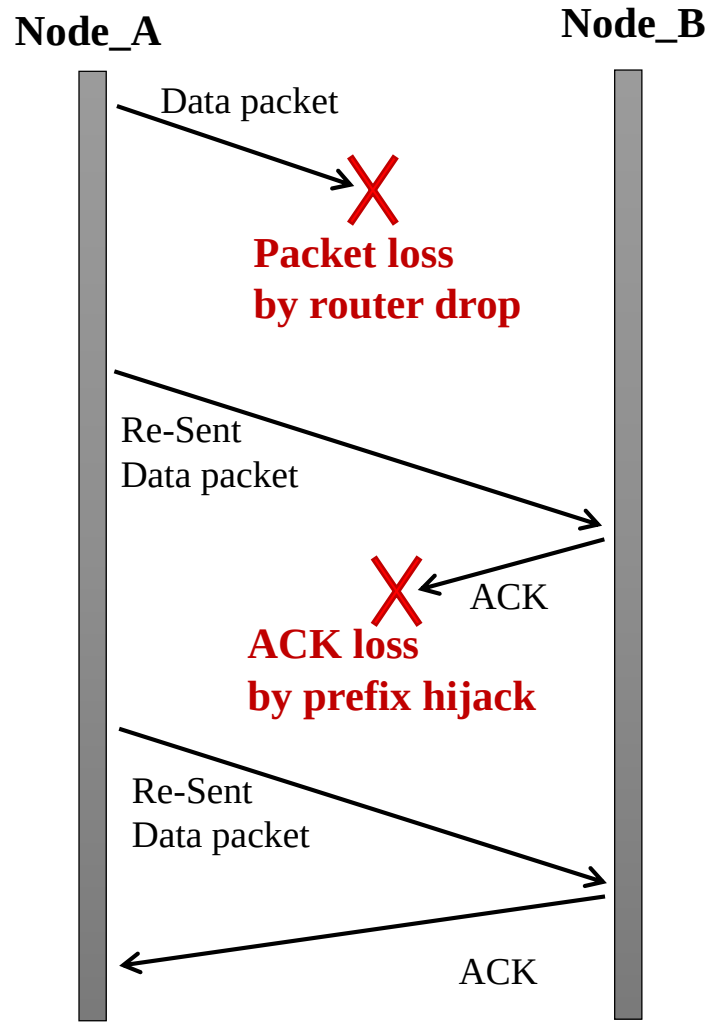
- We seperate the network conditions as:
  - **Synchronous** Network:

    Exist **known** upper bound of network delay

  - **Asynchronous** Network:

    Network delay is not only **unknown** but also **unfixed;**
    Nevertheless, messages can be **eventually** delivered;
    Make ***minimal*** assumptions about the network condition.

Network	Computationally-Secure Robust MPC
Sync.	t<n/2
Async.	t<n/3



TCP (enhanced by TLS)
is Asynchronous !!!

Vulnerabilities of Sync. MPC in Asynchrony

Most (practical) MPCaaS is synchronous !!

- **G.O.D. Violation in Async.**
  - **Excluding those inherent impossible: $t < n$**
  - **Many expected robust MPC:**
    - ✓ **Not *robust* at all in async.**

Protocols	Thld. ($t <$)	IT online	asyn. securities	
			Privacy	GOD
HN06 [56]	$n/2$	✗	✗	✗
GLZ20 [51]	$n/2$	✓	✗	✗
GLS19 [49]	$n/3$	✓	✗	✗
DN07 [41]	$n/3$	✓	✗	✗
BH08 [13]	$n/3$	✓	✗	✗
VIFF [37]	$n/3$	✓	✓	✗
HyperMPC [9]	$n/3$	✓	✓	✗
SPDZ [40, 42]	n	✓	✓	✗
EMP [78]	n	✗	✓	✗

Why cannot wait messages forever to restore robustness ?

Taking BGW as example:

Have to gather all honest parties' shares to correct malicious shares, which is impossible !

That's the challenge of asynchrony !!!

Vulnerabilities of Sync. MPC in Asynchrony

Most (practical) MPCaaS is synchronous !!

- **Privacy Breach in Async.**
  - **Many expected robust MPC:**
    - ✓ ***not private*** **in async.**

Protocols	Thld. ($t <$)	IT online	asyn. securities	
			Privacy	GOD
HN06 [56]	$n/2$	✗	✗	✗
GLZ20 [51]	$n/2$	✓	✗	✗
GLS19 [49]	$n/3$	✓	✗	✗
DN07 [41]	$n/3$	✓	✗	✗
BH08 [13]	$n/3$	✓	✗	✗
VIFF [37]	$n/3$	✓	✓	✗
HyperMPC [9]	$n/3$	✓	✓	✗
SPDZ [40, 42]	n	✓	✓	✗
EMP [78]	n	✗	✓	✗

Many sync. robust MPC “eject” unresponsive parties !

In asynchrony, honest parties can be eventually all “ejected” !

Design Objective

■ Threat model:

- **Full Asynchrony:**
  - ✓ adversary can arbitrarily delay messages
- **Optimal Resilience:**
  - ✓ against t<n/3 malicious corruptions (servers)

■ Security goal:

- **Private** and **G.O.D.**

■ Efficiency requirement:

- almost **I.T. online** phase
- concretely and asymptotically efficient

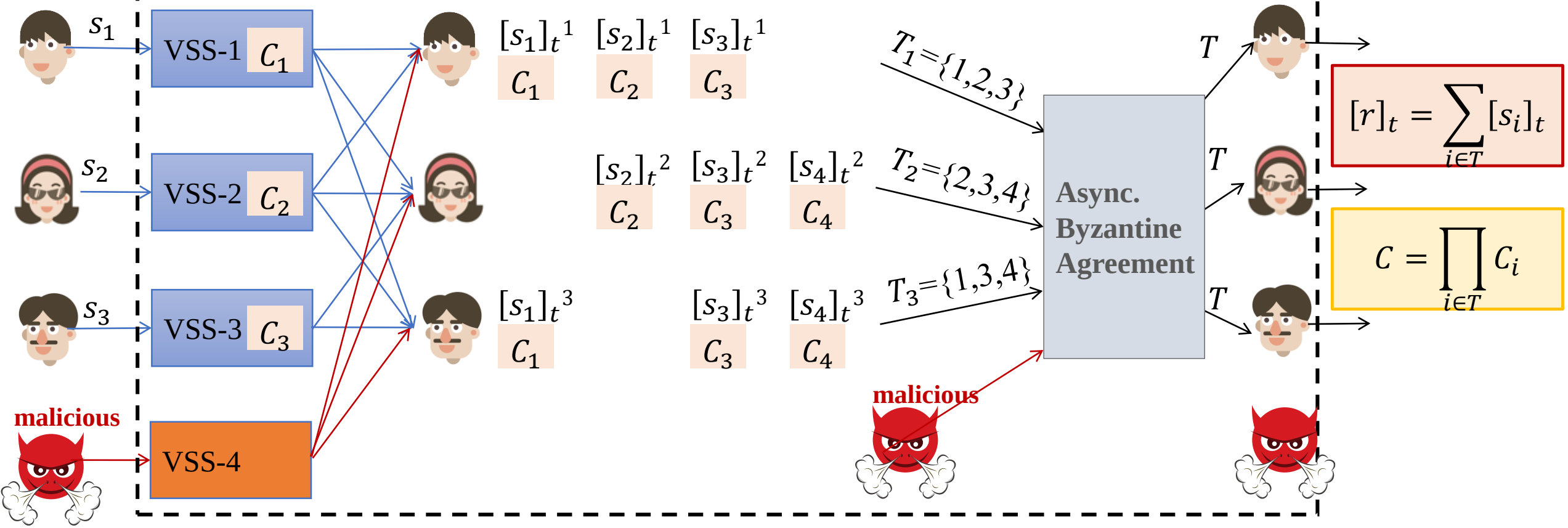
Protocols	Thld. ($t <$)	I.T. online	asyn. G.O.D.	comm. / gate		# secrets to be shared per triple
				Good	Bad	
CHP13 [30]	$n/4$	✓	✓	$O(n)$	$O(n)$	48
CP17 [32]	$n/4$	✓	✓	$O(n)$	$O(n)$	48
DXKR23 [41]‡	$n/3$	✓	✗	$O(n^3)$	–	$2n$
	$n/4$	✓	✓	$O(n^3)$	$O(n^3)$	$2n$
hbMPC [66]	$n/3$	✓	✗	$O(n)$	–	12
HNP08 [60]	$n/3$	✗	✓	$O(n^2)$	$O(n^2)$	-
CP15 [31] †	$n/3$	✗	✓	$O(n)$	$O(n)$	-
CHL21 [28]	$n/3$	✗	✓	$O(n^2)$	$O(n^2)$	-
PCR08 [60]	$n/3$	✓	✓	$O(n^5)$	$O(n^5)$	$9n+6$
CP23 [33]	$n/3$	✓	✓	$O(n^4)$	$O(n^4)$	$6n$
SS23 [77]	$n/3$	✓	✓	$O(n^2)$	$O(n^3)$	$6n$
GS23 [54]	$n/3$	✓	✓	$O(n^2)$	$O(n^3)$	$3n$
GLS24[52]	$n/3$	✓	✓	$O(n)$	$O(n)$	$\sim 230K$
Dumbo-MPC	$n/3$	✓	✓	$O(n)$	$O(n^2)$	$n+6$

Prior Art & Remaining Challenges

GRR98/GS23

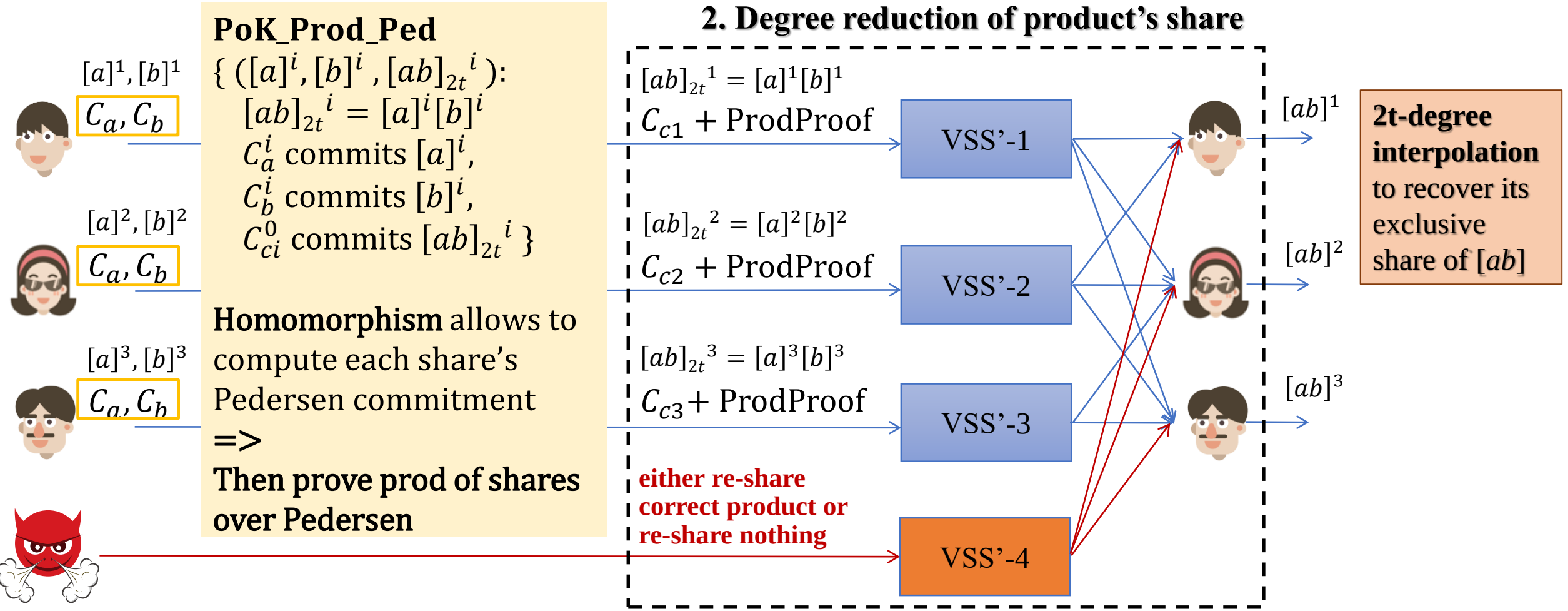
- **Random sharing + degree reduction + Pedersen polynomial commitment + proof-of-product over Pedersen**

1. Generation of random sharing



GRR98/GS23

- **Random sharing + degree reduction + Pedersen polynomial commitment + proof-of-product over Pedersen**



GRR98/GS23

- **Random sharing + degree reduction**
  + Pedersen polynomial commitment + proof-of-product over Pedersen

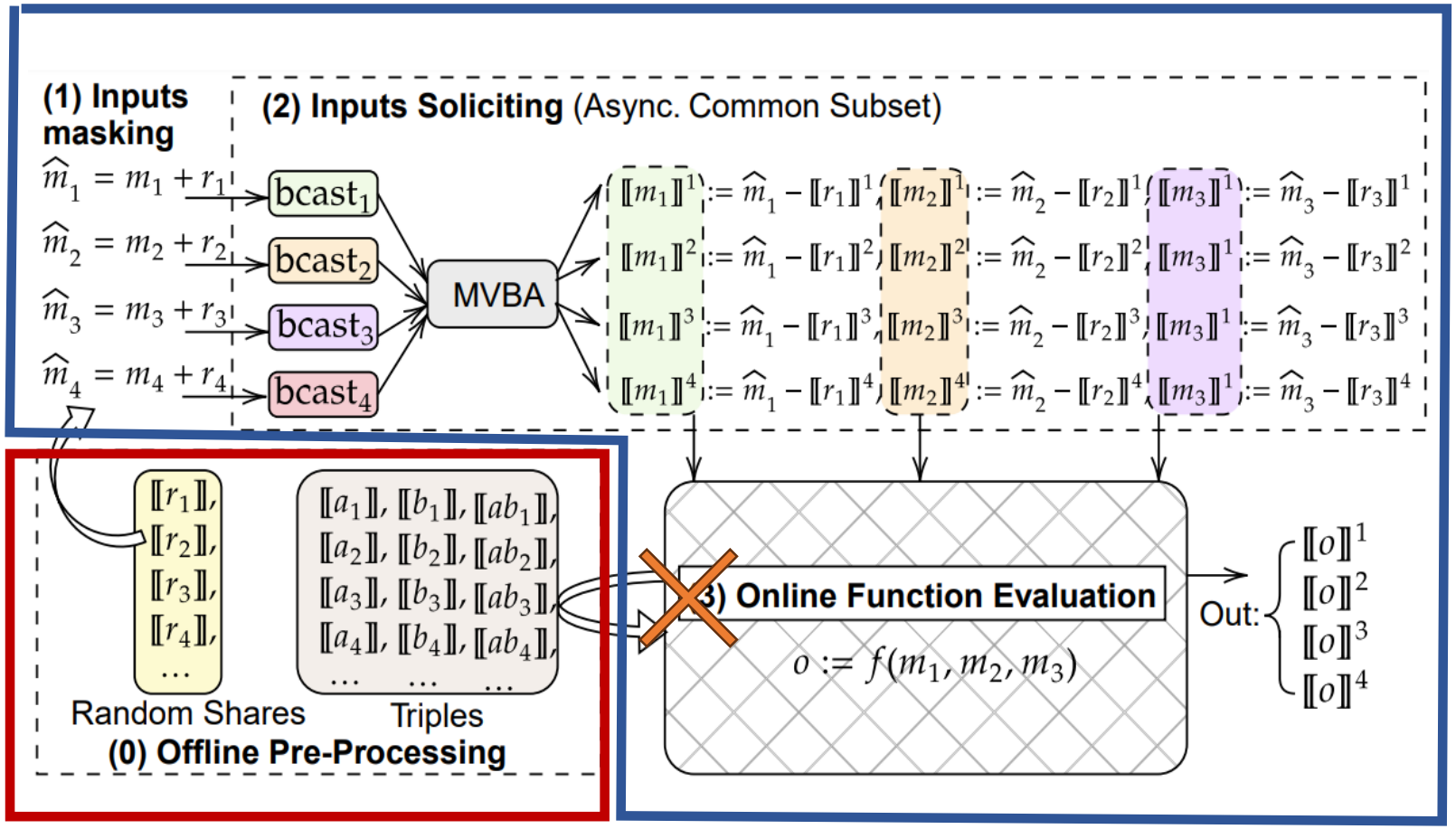
- **Remaining efficiency issues:**
  - Worst-case ***cubic*** communication complexity per triple
    (due to large size of Pedersen commitment)
  - Require ***3n*** secrets to be shared per triple
    (due to inefficient generation of random sharing)

HoneyBadgerMPC: sacrifice offline robustness

- **Robust online phase + non-robust offline phase** (requiring all parties honest)

  => **no G.O.D.,** but can generate triples much faster than GS23 in very good case

Non-robust offline eventually means non-robust online



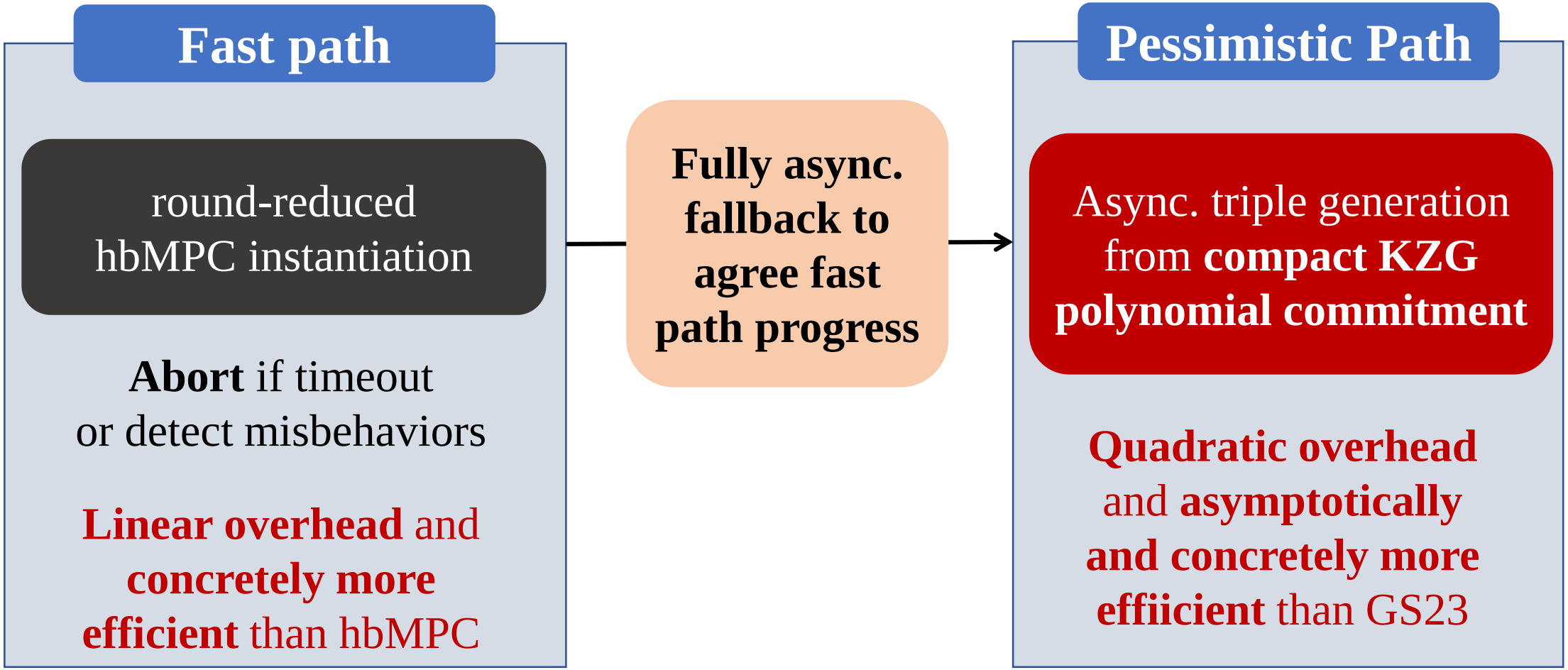
Recap the state-of-the-art

- **GS23:** all-phase **robust**, but very **inefficient**
- **hbMPC:** offline **non-robust**, but **more efficient in good case**

- **Question: Can we have the best-of-both ?**
  - **Bad case:** both asymptotically and concretely better than GS23
  - **Good case:** at least as high performant as hbMPC and
    can securely fallback when optimistic condition fails

Our Solution: Dumbo-MPC

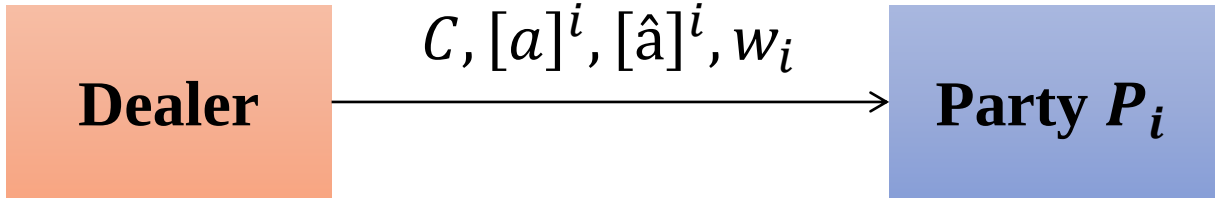
High-level overview



Proof of secrets' product over KZG

■ Devise proof-of-product relation over KZG commitment

- why Pedersen convenient => interpolate in exponent to derive any evaluation's commitment
- Augment KZG commitment by introducing **hidden evaluation interface**



New interface computes:

$$T_i = g^{[a]^i} h^{[\hat{a}]^i}$$

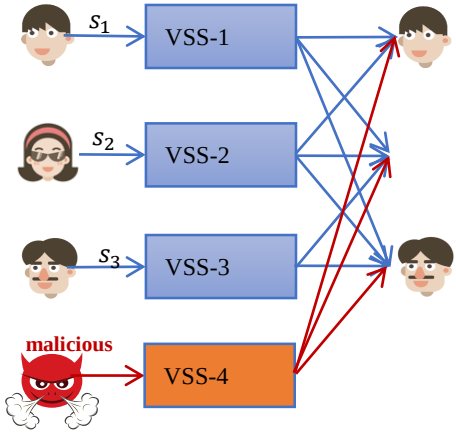
-and-

ω_i binds T_i to C

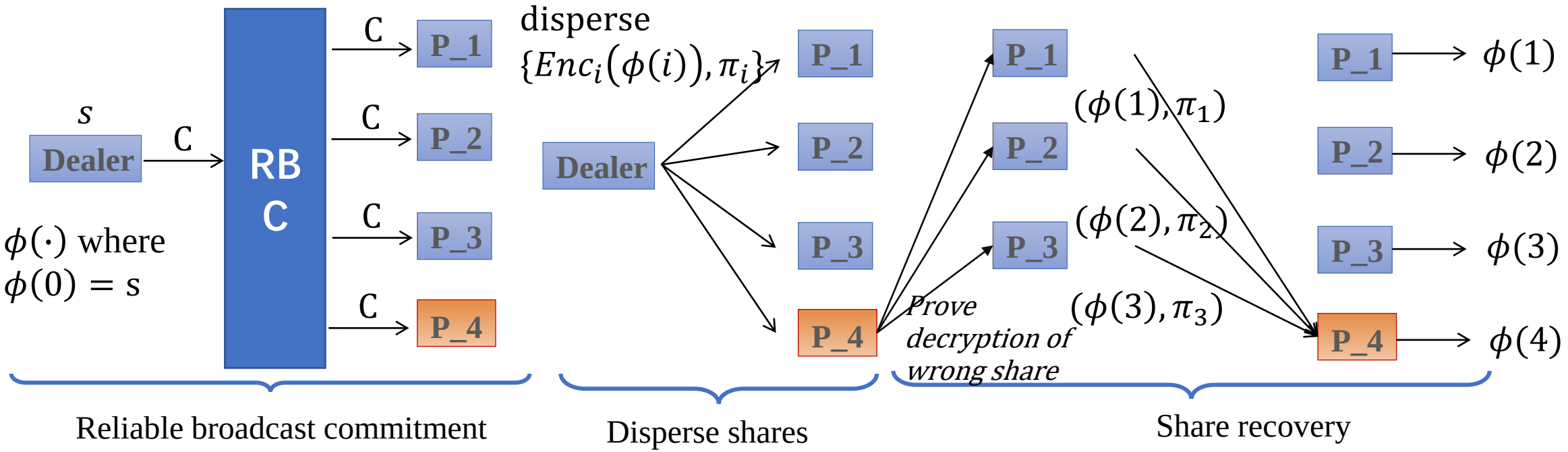
- Hidden evaluations are:
  - ✓ **Binding** => cannot open as secrets not binding to C
  - ✓ **Hiding** => all $\{T_i\}$ and all $\{\omega_i\}$ do not leak extra information

- Then, can prove correct multiplication by zk-PoK of product relation over Pedersen

Efficient AVSS instantiation w/ concurrent composability

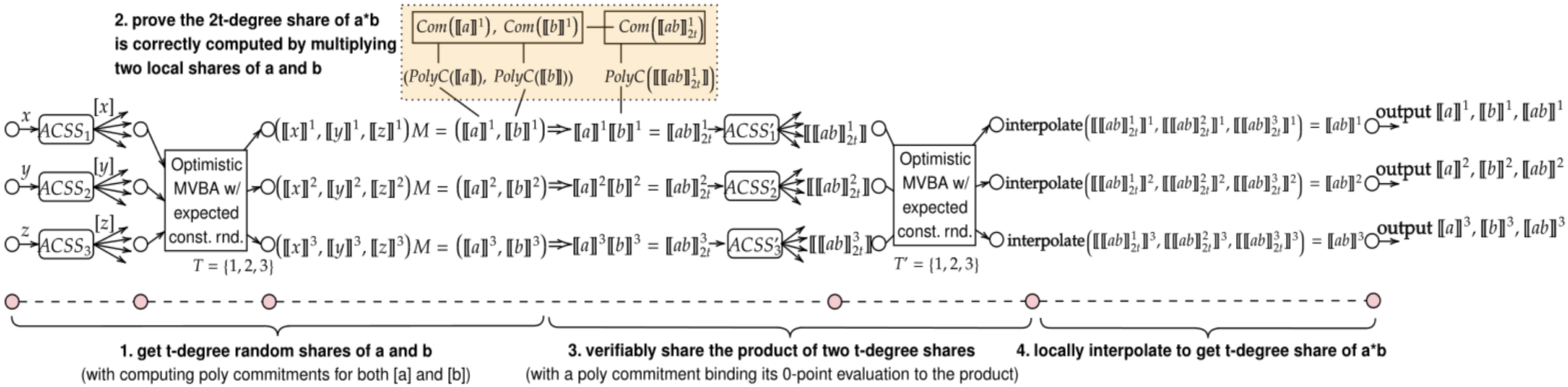


- Using KZG to instantiate the state-of-the-art **batch AVSS** (hbACSS)
- **Remaining non-triviality: lacking concurrent composability in hbACSS**
  - drop-in IND-CCA PKE + using different keys in different dealer's instances



More efficient triple generation protocol

- **Efficient triple instantiation from compact KZG commitment:**
  - **Asymptotically more efficient AVSS** for secret sharing (as succictness of KZG)
  - Good-case **round-reduced** async. Byzantine agreement (**good-case terminate in 5 rounds**)
  - Extract random sharing via super-invertible matrix (as homomorphism of KZG) => **reducing # of secrets to share from *3n* to *n*+O(1)**

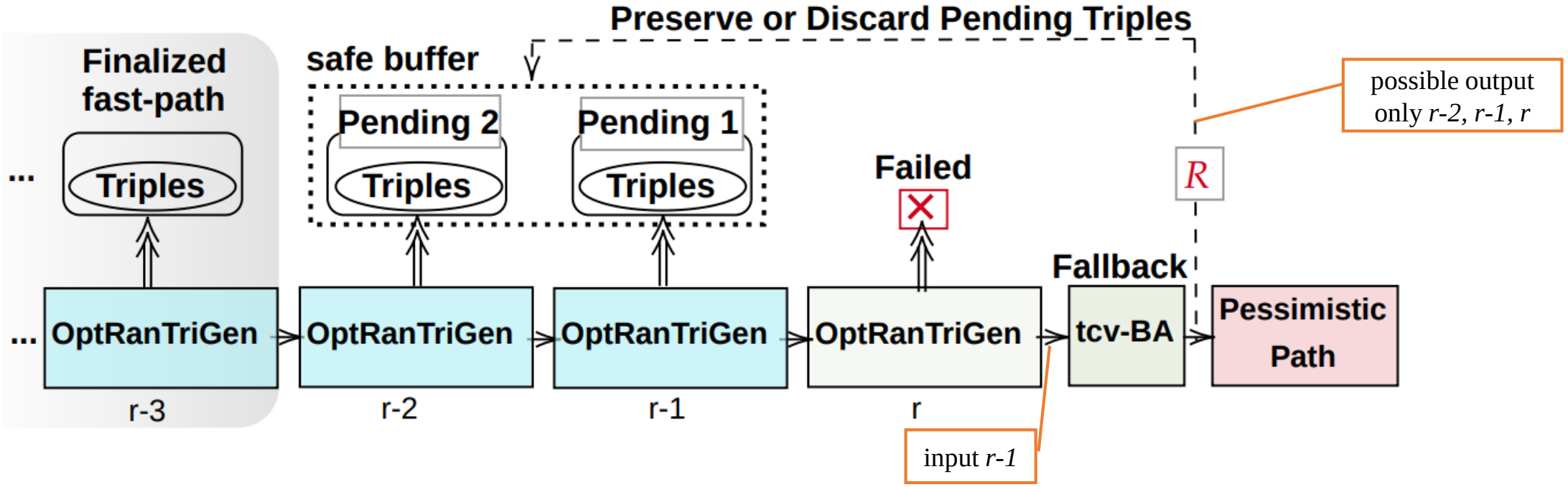


Dual-mode triple generation framework

■ Fast Path and Fallback:

- **Fast-path features:** when abort in round *r*, all honest parties must finish round *r-2*
  => all triples before round *r-2* can be finalized

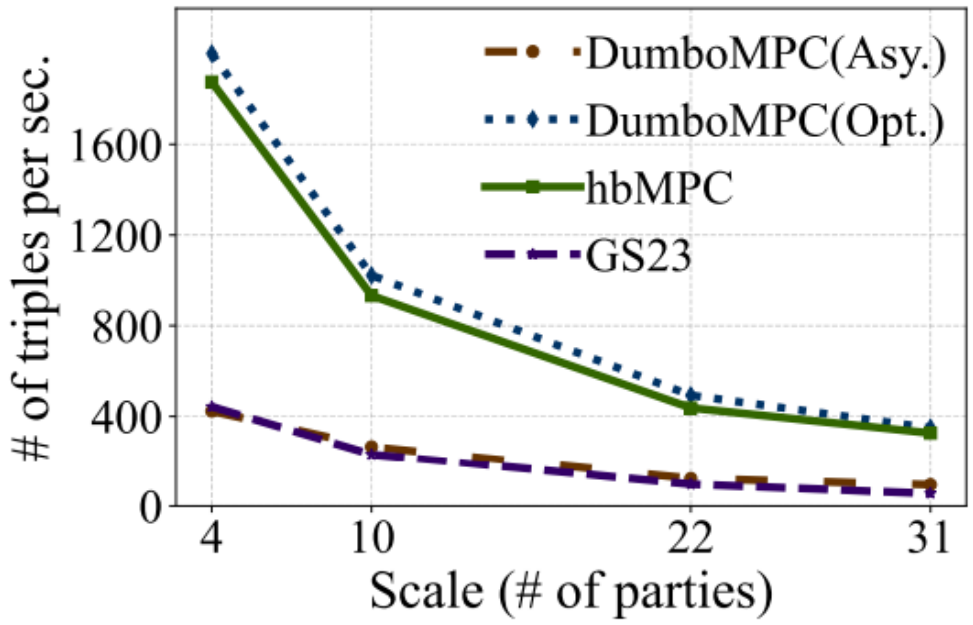
- **Fallback implementation:** two-consective-value Byzantine agreement
  => conceptually minimal BA (choose one out of two integers)



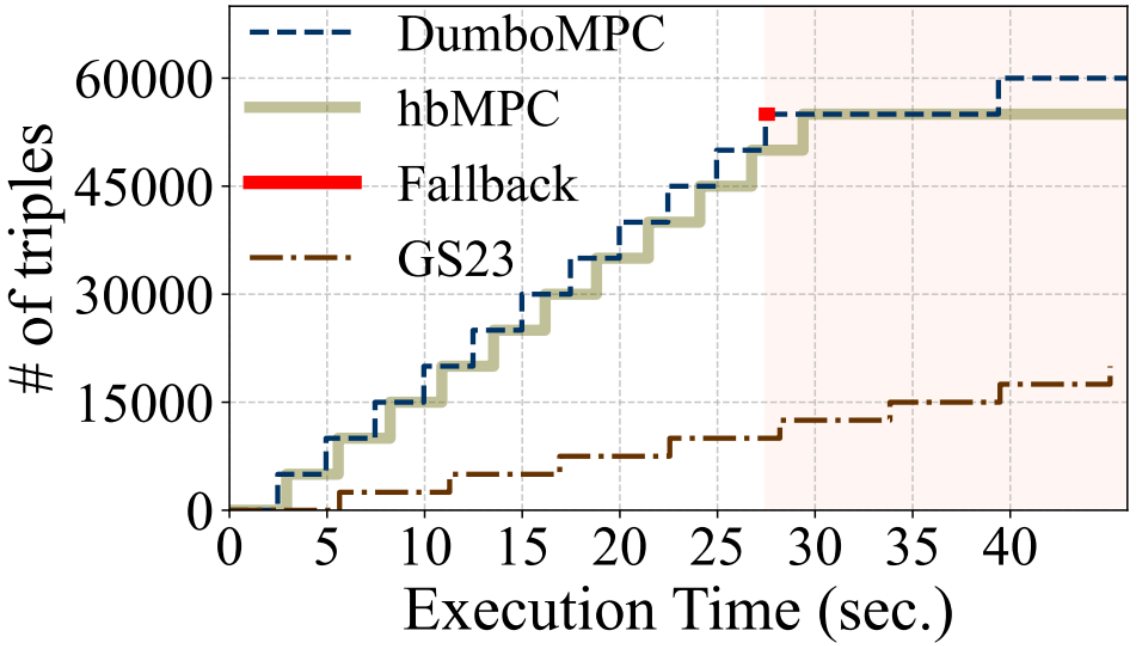
Evaluations

- **Bad case faster than GS23**
- **Good case faster than hbMPC + secure fallback**

triple throughput for varying *n*



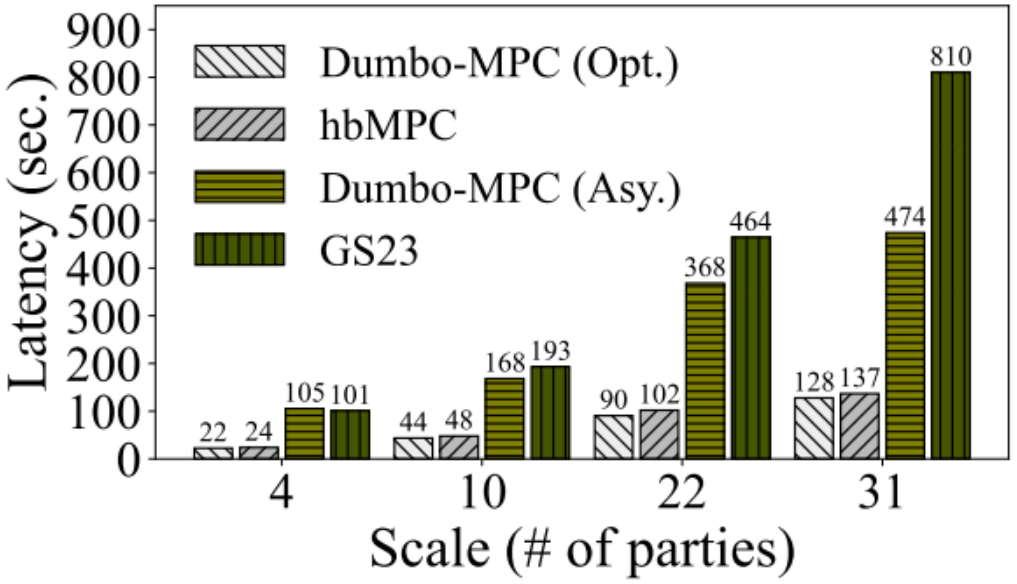
***n=4* in face of a dynamic crash**



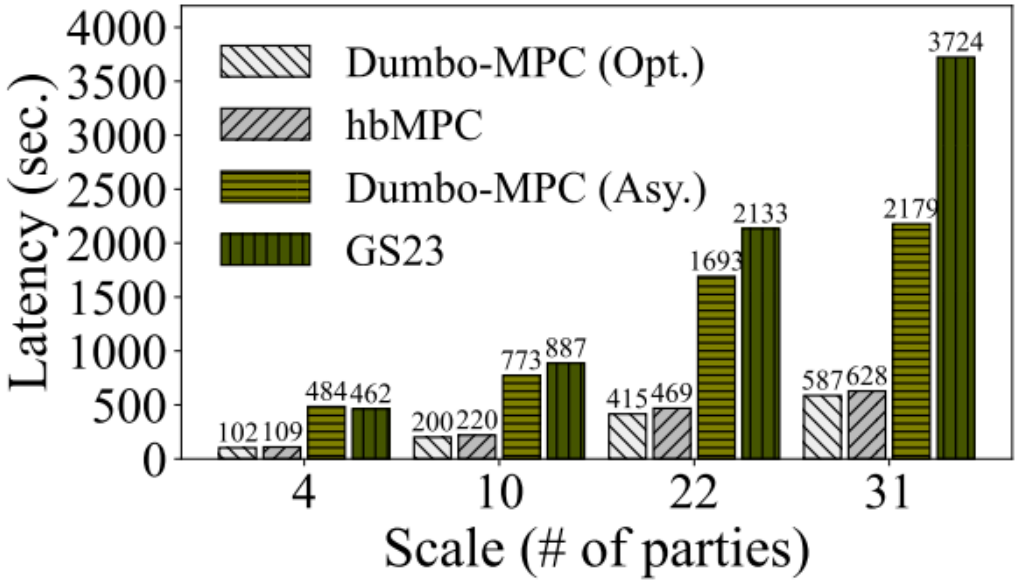
Evaluations

- **Bad case faster than GS23**
- **Good case faster than hbMPC + secure fallback**

Latency of pre-processing triples for (a) Vickery auction and (b) mixing net



(a) Vickery auction



(b) Switching network

Summary

Async. MPC-as-a-Service framework Dumbo-MPC (BGW paradigm from SS)**:**

- **More efficient async. triple generation**
  - ✓ efficient instantiation as proof-of-product over KZG
  - ✓ asymptotically and concretely better than GS23
- **Good-case fast path + secure async. fallback**
  - ✓ slightly faster than hbMPC by reducing redundant rounds
  - ✓ Secure async. fallback

Thanks !