More is Merrier: Relax the Non-Collusion Assumption in Multi-Server PIR

Tiantian Gong[1], Ryan Henry[2], Alexandros Psomas[1], Aniket Kate[1,3]

[1]Purdue University, [2]University of Calgary, [3]Supra Research

@PETS Workshop in PIR, Jul 15, 2024

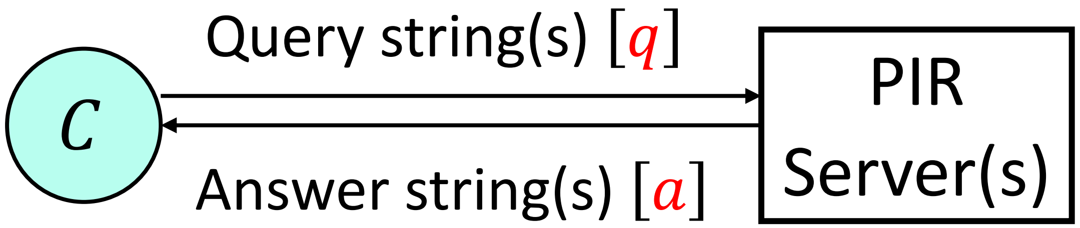
Plan

1. Motivation
2. Progressively build up to the solution
3. Overhead
4. Future directions

Private Information Retrieval (PIR) [CKGS95]

To query a database $\mathbb{D}$ with n entries:

$[q] = Query(x)$



$a_i = Compute\ (\mathbb{D}, q_i)$

Client's index of interest: $x \in [n]$

$\mathbb{D}_x = Reconstruct\ ([a])$

Security

- Correctness – C can reconstruct $\mathbb{D}_x$:

$$\mathrm{H}(X | a_1 = \mathrm{Compute}(\mathbb{D}, q_1), \ldots, a_k = \mathrm{Compute}(\mathbb{D}, q_k)) = 0$$

- ***t*-Privacy** (IT, computational) – less than $(\boldsymbol{t} + \mathbf{1})$ parties learn no extra info:

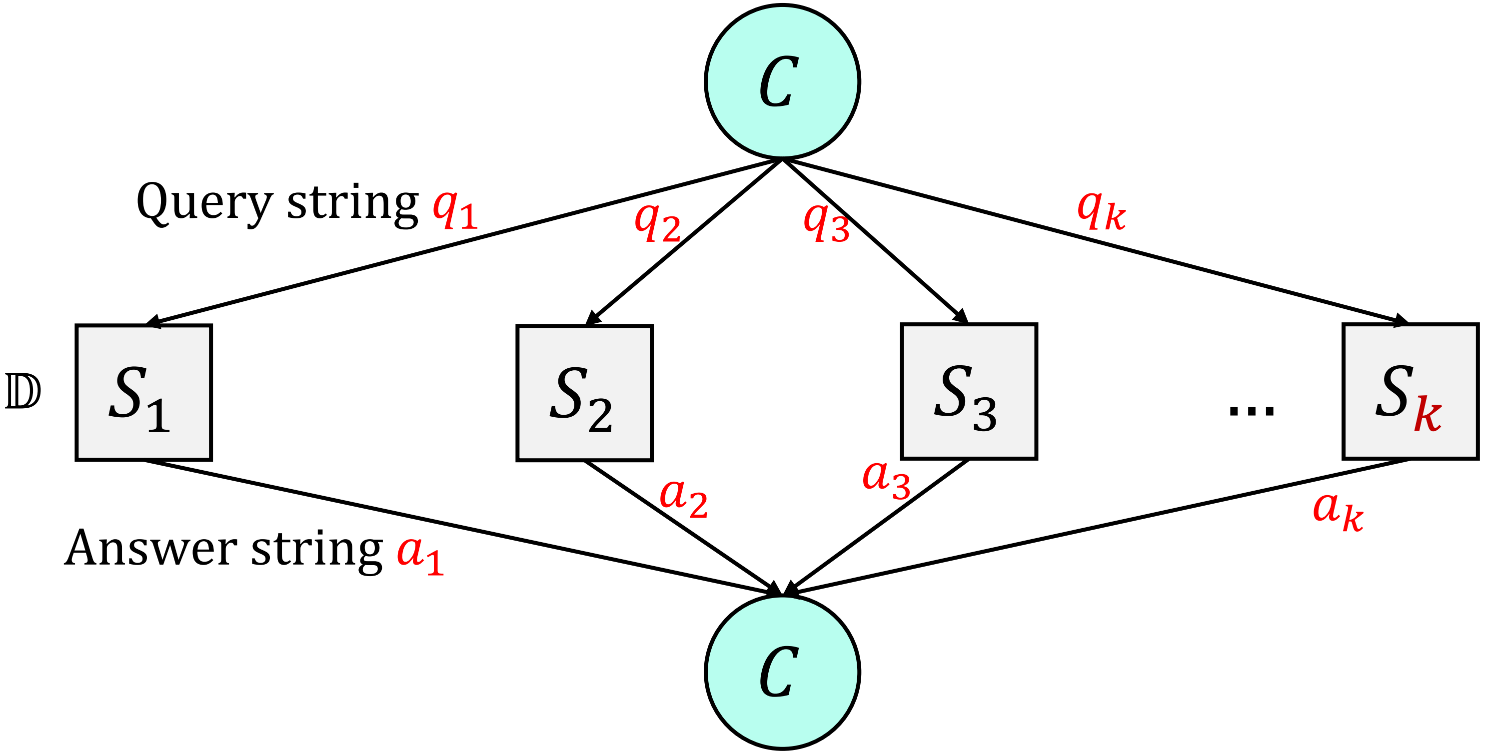
$$\mathrm{H}\left(X | \{q_j\}_{j \in S, |S| \le t}\right) = \mathrm{H}(X)$$

$\mathrm{H}(\cdot)$ computes the entropy of a random variable; X is the random variable for x

The Collusion Problem in Multi-Server PIR

PIR Constructions

- Single-server ($k = 1$)
- **Multi-server ($k \geq 2$)***



***Our focus**

k-out-of-k t-private PIR: k responses to reconstruct & $t + 1$ servers can learn extra info

$\ell \geq k$ servers

More efficient but t-private multi-server PIR assumes *at most t severs collude

! More servers can **easily** collude over **unobserved communication channels** to learn x – which is **impossible** to detect

Relax the non-collusion assumption

Setting:

1. Servers can collude over *unobserved* channels with *any protocol* to learn about user secret x
2. After *successful* collusion, at least some colluding parties have learned something nontrivial about the secret x --- denote as $f(x)$

Relax to rationality assumption, i.e., servers are either rational or malicious

Goal

an **algorithm** on a public bulletin board

Design a mechanism such that

(a) it induces a ***game*** where *exactly* one of the servers can take advantage of the nontrivial ***information gain*** $f(x)$ to maximize its utility at the expense of others,

(b) resulting in some parties ***unwilling*** to collude to give other servers such an advantage

Measure non-trivial information gain

X: r.v. for secret x on finite alphabet $\mathcal{X}$. Consider

$$f: \mathcal{X} \mapsto \mathcal{Y}$$

We call $f(\cdot)$ γ-nontrivial if for some parameter $\gamma \in (0,1)$

$$\mathbb{P}[\text{guess } f(x) \text{ correctly}] \leq \gamma$$

- Naively, evaluate f at all inputs and compute the probability of the most likely output
- $f(\cdot)$ is bijective

$$\mathrm{H}(f(X)) = \mathrm{H}(X)$$

- $f(\cdot)$ is injective, utilize the lower bound for the ***entropy of a function on a r.v.*** [Sason18]

Thought experiment

Simple Mechanism M_0

Unknown: secret $f(x)$

Known: secret worth V, server set $\{S_1, \dots, S_{\ell=k}\}$

▷ **Winner selection rule W_0**

If server S_1 tells M_0 the correct secret, select S_1 as *winner* and mark all other parties as *colluders*

S_1

S_2 S_3 … S_k

▷ **Payment rule P_0**

(1) Reward the *winner* amount $\lambda_r > 0$;

(2) Penalize each marked *colluder* amount $\lambda_p > V$; (realized via deposits)

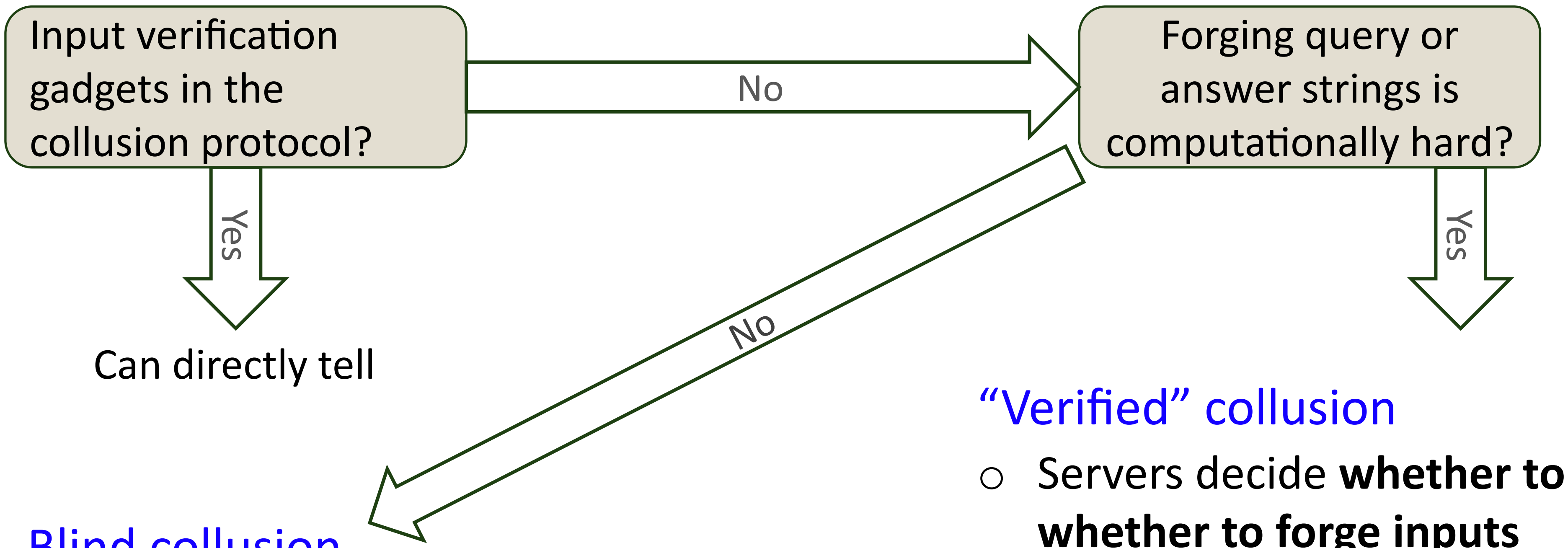
(3) Penalize S_1 amount λ_p if it tells a wrong secret

Analysis **rational** servers, **single** run

- S_1 is incentivized to signal collusion after successful collusion
- Others are not incentivized to let S_1 learn the secret

1) How can S_1 tell if collusion is successful <- may only receive output
2) Nothing is stopping S_1 from helping others learn the secret
3) S_1 can have arbitrary prior knowledge about the secret -> framing others is possible

1) Tell a successful collusion



"Verified" collusion

- Servers decide **whether to collude,** and **whether to forge inputs**
- One can tell the collusion is **successful** with negligible error --- check if the output is gibberish

Blind collusion

- Servers decide **whether to collude,** and **whether to forge inputs**
- One *cannot* tell if collusion is successful with negligible error by examining the outputs

2) Stop S_1 from helping others learn

Still Simple Mechanism M_1

Unknown: secret $f(x)$

Known: secret worth V, server set $\{S_1, \dots, S_{\ell=k}\}$

▷ **Winner selection rule W_1**

If any server S_i tells M_1 the correct secret *first*, select S_i *as winner* and mark all other parties as *colluders*

S_i S_{-i}

▷ **Payment rule P**

(1) Reward the *winner* amount $\lambda_r \geq 0$;

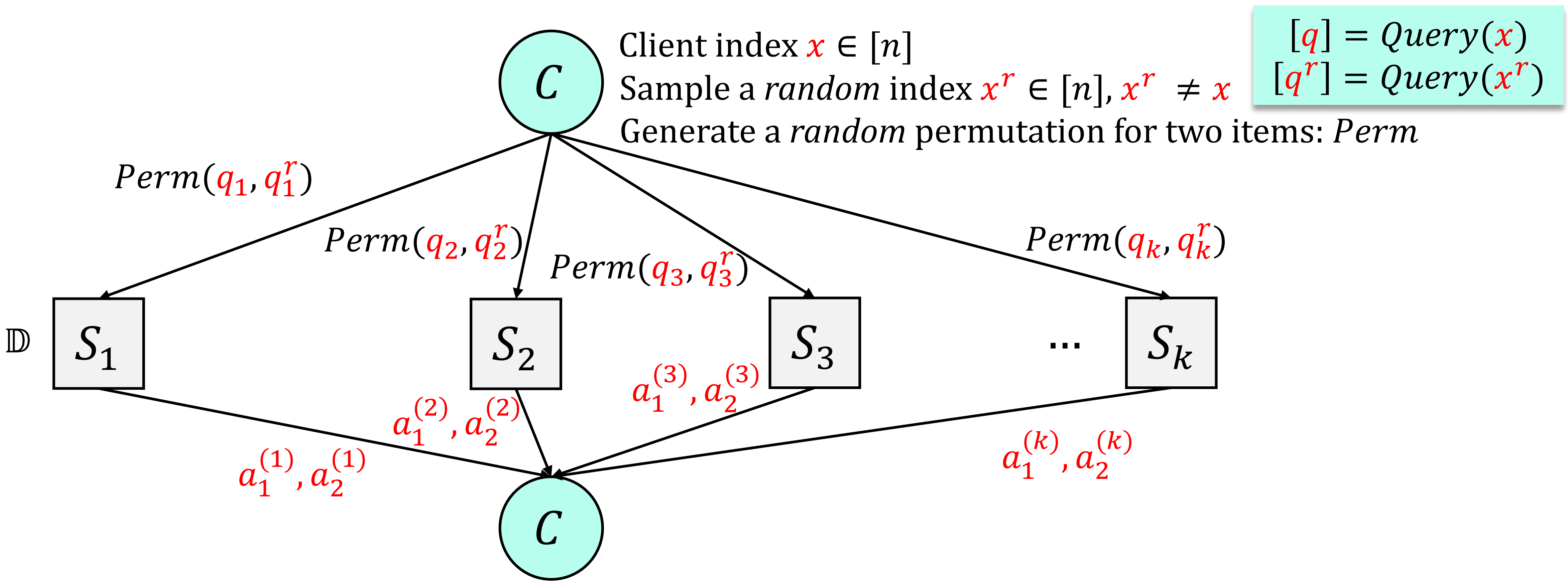
When there exists competition in telling the secret, we do *not* need positive rewards.

(2) Penalize each marked *colluder* amount $\lambda_p > V$;

(3) Penalize a server amount λ_p if it tells a wrong secret

3) Accommodate arbitrary private knowledge

by generating $\boldsymbol{\omega} \geq \mathbf{1}$ random companion queries



Example with $\omega = 1$

Updated mechanism

Mechanism $\boldsymbol{M_2}$

Unknown: secrets $f(x), f(x^1), \dots, f(x^\omega)$

Known: secret worth V, server set $\{S_1, \dots, S_{\ell=k}\}$

▷ **Winner selection rule $\boldsymbol{W_2}$**

If any server S_i tells M_2 a correct secret first along with its *corresponding input*, select S_i *as winner* and mark all other parties as *colluders*

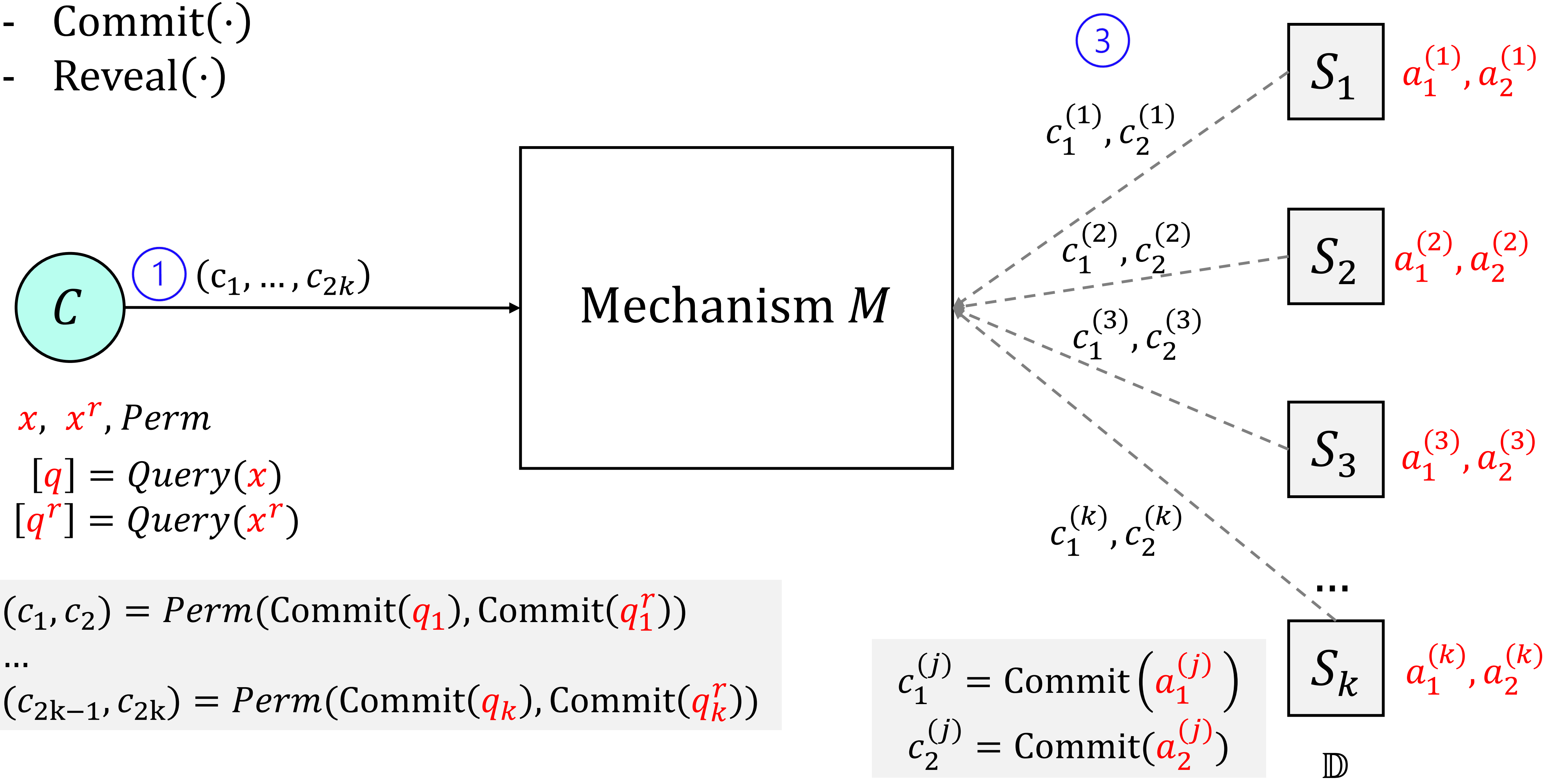
▷ **Payment rule $\boldsymbol{P}$**

4) How to verify the report of the secret?
5) Client collusion?
6) What are the exact payment amounts so that we achieve the non-collusion outcome? Will the amounts be practical?

4) Verify reports - Setup

Assume a secure[1] commitment scheme:

- Commit(·)
- Reveal(·)



$x,\ x^r, Perm$

$[q] = Query(x)$

$[q^r] = Query(x^r)$

$(c_1, c_2) = Perm(\mathrm{Commit}(q_1), \mathrm{Commit}(q_1^r))$

…

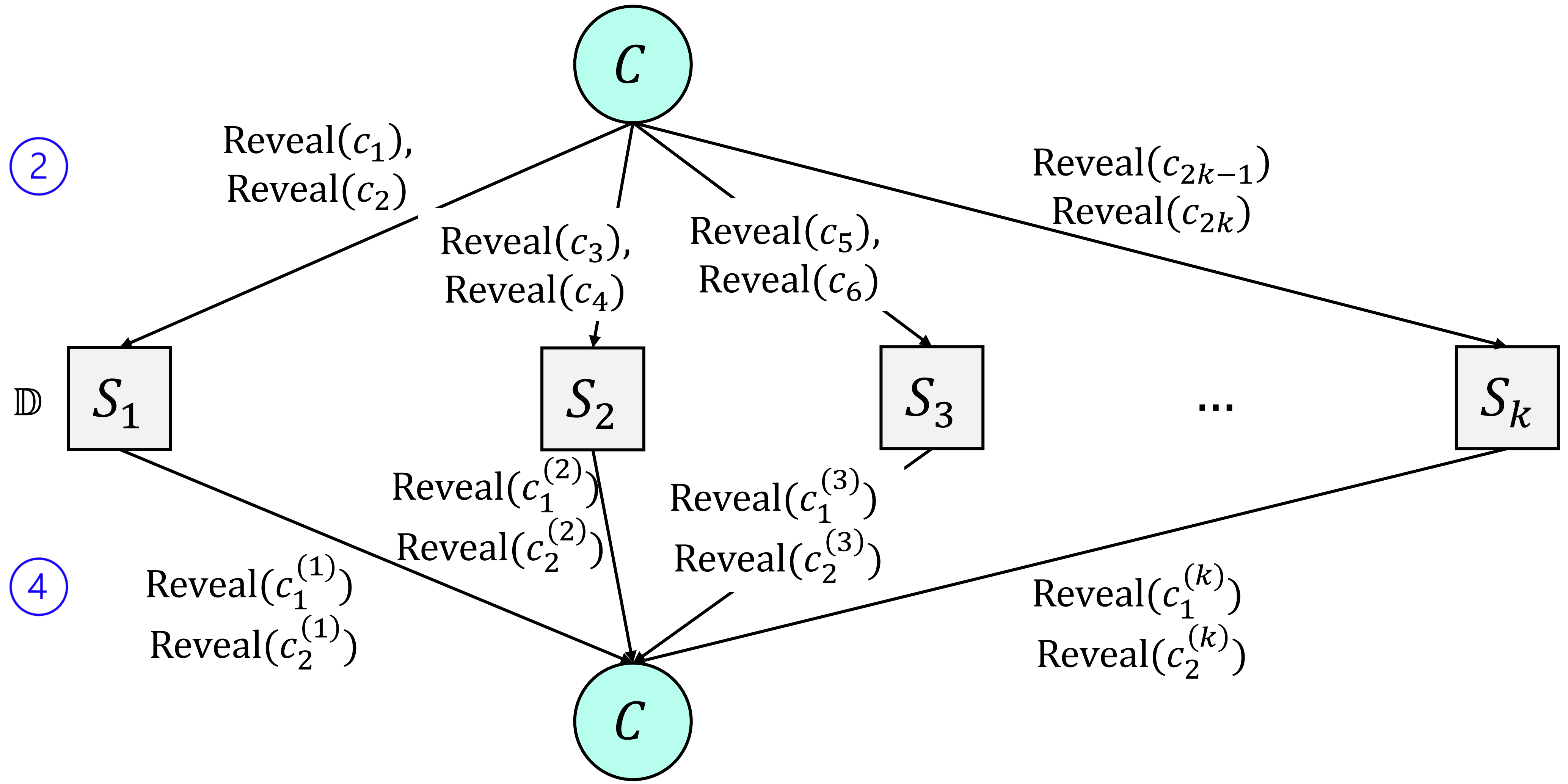
$(c_{2k-1}, c_{2k}) = Perm(\mathrm{Commit}(q_k), \mathrm{Commit}(q_k^r))$

$c_1^{(j)} = \mathrm{Commit}\left(a_1^{(j)}\right)$

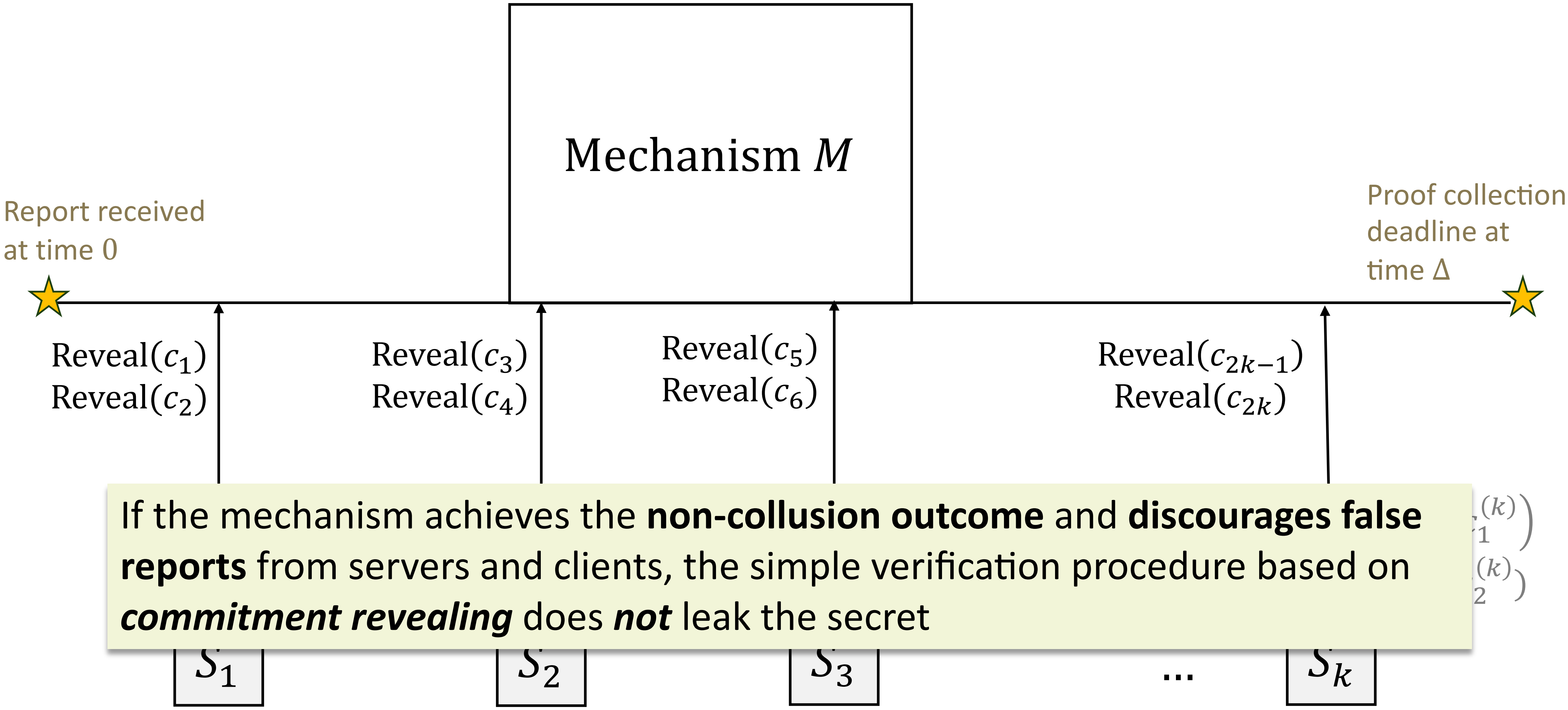
$c_2^{(j)} = \mathrm{Commit}(a_2^{(j)})$

[1] We desire perfectly hiding, computationally binding commitment

4) Verify reports - Setup



4) Verify reports - 0 malicious parties



If the mechanism achieves the **non-collusion outcome** and **discourages false reports** from servers and clients, the simple verification procedure based on ***commitment revealing*** does ***not*** leak the secret

5) Client collusion

+ Charge service fees λ_s for each queried server from the client.

By having

$$(k-1)\lambda_s > \lambda_r$$

We can disincentivize client collusion.

6) Parameterize payments – Desired outcome

▷ **Payment rule P**

(1) Reward the *winner* amount $\lambda_r \geq 0$;

(2) Penalize each marked *colluder* amount $\lambda_p > V$;

(3) Penalize a server amount λ_p if it tells a wrong secret;

(4) Charge service fees λ_s for each queried server from the client and transfer to servers if there is no collusion after a privacy protection window

Desired outcome $O^{\star}$: In equilibrium, servers do not successfully collude

6) Parameterize payments – Achieve $O^\star$ in equilibrium

Theorem 1 (Informal, $\ell \geq k$) rational servers, single run

In a single run of the ℓ-party collusion game, $O^\star$ is achieved when $\lambda_p > 0$ and $\lambda_s + \frac{k-1}{k}\lambda_p > V$.

*$\lambda_r = 0$

*$(k-1)\lambda_s > \lambda_r \Rightarrow \lambda_s > 0$ (discourage client collusion)

*reminder: a client picks k servers *at random* to send queries to if $\ell > k$

Corollary 1 (Informal , $\ell \geq k$) rational servers

In known finite runs of the ℓ-party collusion game, $O^\star$ is achieved when $\lambda_p > 0$ and $\lambda_s + \frac{k-1}{k}\lambda_p > V$.

Q. What about infinite *or* unknown runs?

6) Parameterize payments – Achieve $O^\star$ in equilibrium in repetition

What is special about unknow or infinite number of runs:

- **Folk theorem** says that if players are ***patient*** enough, **any payoff** can appear in equilibrium
- In **well-studied games** (e.g., prisoner's dilemma), infinite repetition brings *multiplicity of equilibria* (hard to make predictions)

6) Parameterize payments – Achieve $O^\star$ in equilibrium in repetition

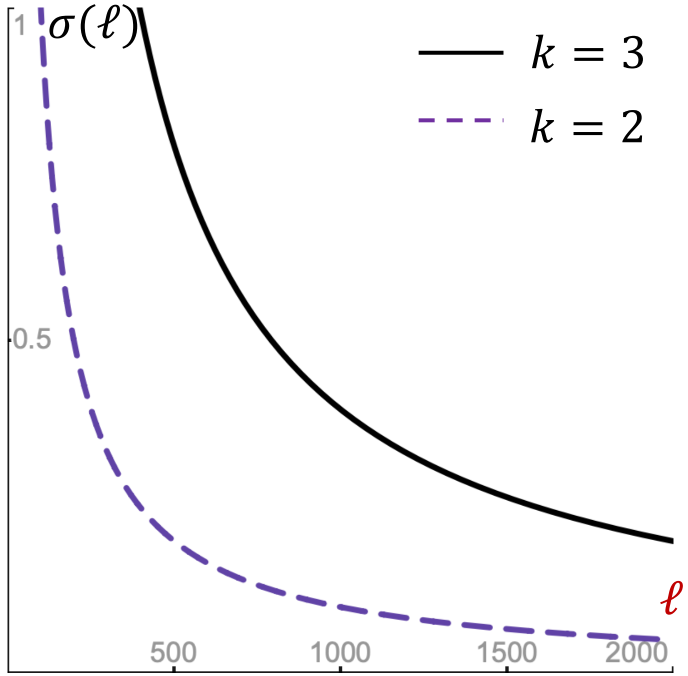
Theorem 2 (Informal, $\ell \gg k$) rational servers

In unknown or infinite runs of the ℓ-party collusion game, $O^\star$ is achieved when $\sigma(\ell)\, V < \boldsymbol{\lambda_r} \le \lambda_p$ and $\lambda_s + \lambda_p > \frac{k-1}{k}(\lambda_r + \lambda_p) + V$ where $\sigma(\ell)$ decreases with ℓ

*a larger ℓ allows a larger parameter feasibility region,

hence "More is Merrier"

*an alternative: replace players periodically ⇒ finite runs



Proposition 1 [Existence of solution] (Informal)

Practical parameters satisfying Theorem 1 or 2 always exist.

Add malicious parties

Update 1: Parameterization

Update 2: Report verification

Corollary 2 (Informal, $\ell \geq k$)

In a single or a known finite runs of the ℓ-party collusion game **with $\boldsymbol{k-2}$ adaptive malicious corruptions**, $O^\star$ is achieved when $\lambda_p > 0$ and $\lambda_s + \frac{1}{2}\lambda_p > V$.

In unknown or infinite runs of this game, $O^\star$ is achieved when $\frac{\delta}{1-\delta}(1-q)V < \lambda_r \leq \lambda_p$ and $\lambda_s + \lambda_p > \frac{1}{2}(\lambda_r + \lambda_p) + V$.

Add malicious parties – Parameterization for static corruption

Corollary 3 (Informal, $\ell > k$)

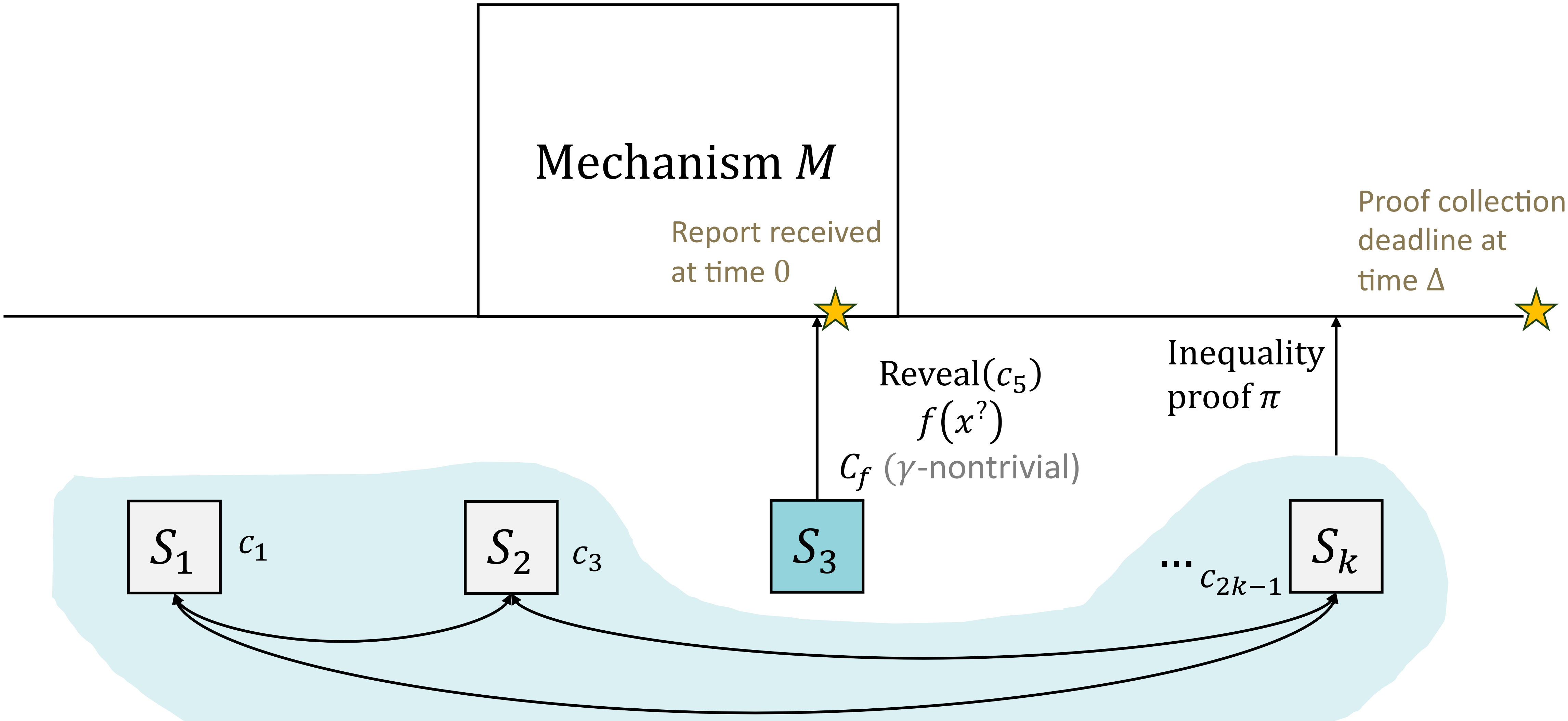
In a single run or known finite runs of the ℓ-party collusion game **with $\boldsymbol{\theta\ell}$ static malicious corruptions**, with probability $1 - 2^{-\eta}$, $O^\star$ is achieved when $\lambda_p > 0$ and $\lambda_s + \frac{1}{2}\lambda_p > V$ where θ satisfies

$$\frac{\binom{(1-\theta)\ell}{0}\binom{\theta\ell}{k}}{\binom{\ell}{k}} + \frac{\binom{(1-\theta)\ell}{1}\binom{\theta\ell}{k-1}}{\binom{\ell}{k}} \leq 2^{-\eta}$$

Corollary 4 (Informal, $\ell \gg k$)

In infinite runs of the ℓ-party collusion game **with $\boldsymbol{\theta\ell}$ static malicious corruptions** where θ satisfies the above condition, with probability $1 - 2^{-\eta}$, $O^\star$ is achieved when $\frac{\delta}{1-\delta}(1-q)V < \lambda_r \leq \lambda_p$ and $\lambda_s + \lambda_p > \frac{1}{2}(\lambda_r + \lambda_p) + V$.

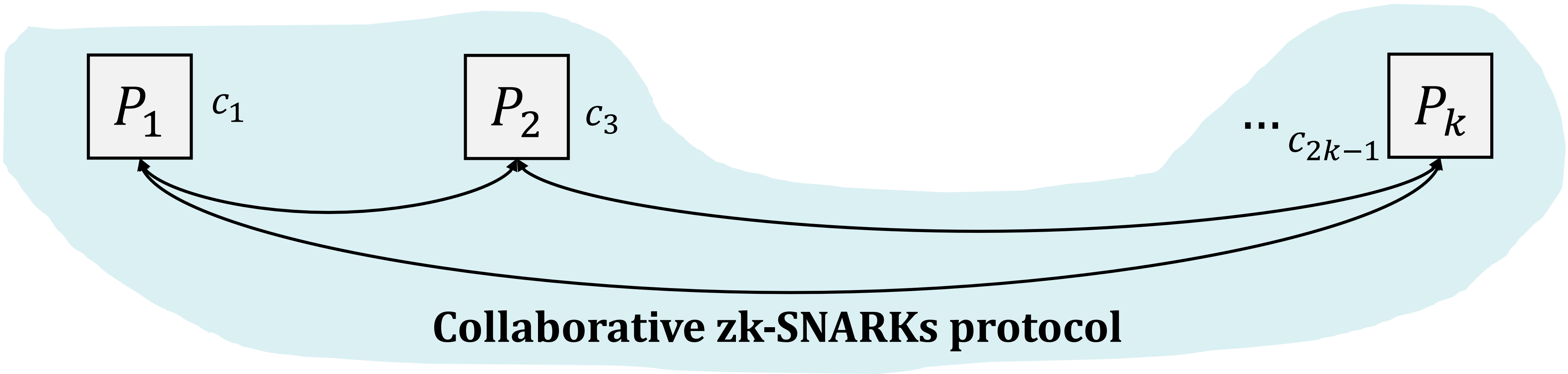
Add malicious parties – Verify reports



Collaborative zk-SNARKs protocol (MPC+zk) [OB22]*

*For $k = 2$, zk-SNARKs protocol like [Groth16], Plonk [GWC19] can be adopted

Add malicious parties – Verify reports



What to prove

Either show that the function $f(\cdot)$ is **trivial** or prove that

1. The **inputs** are correct with respect to the corresponding commitments
2. The **function** $f(\cdot)$ is being computed
3. The **output** is not the value specified in the report

Mechanism overview

Mechanism *M*

Unknown: secrets $f(x), f(x^1), \ldots, f(x^\omega)$

Known: secret worth V, server set $\{S_1, \ldots, S_\ell\}$

▷ **Winner selection rule *W***

If any server S_i tells M the correct secret first along with its *input* and *a proof of inequality is not provided by time* Δ, select S_i *as winner* and mark all other parties as *colluders*

▷ **Payment rule *P***

(1) Reward the *winner* amount $\lambda_r > 0$;

(2) Penalize each marked *colluder* amount $\lambda_p > V$;

(3) Penalize S_1 amount λ_p if it tells a wrong secret;

(4) Charge service fees λ_s for each queried server from the client and transfer to servers if there is no collusion after a privacy protection window

Communication and computation overhead

On paper

One additional commitment per message – instantiated with SHA-3 (or Pedersen commitment when there exist malicious servers)

Implementation as a smart contract on Ethereum

CheckCircuits(·) checks if the function is trivial with oracle services

Table 1. Gas costs summary

Normal service	Gas	Dollars	Collusion resolution	Gas	Dollars
Contract deployment	4697299	\$8.63	Accuse(·)	223766	\$0.41
Deposit(·)	105436	\$0.19	CheckCircuits(·)	66991+	\$0.12+
PostRequests(·)	405657	\$0.74	VerifyExchange(·)	61822	\$0.11
SubmitResponse(·)	97400	\$0.18	VerifyGeneralFunc(·)	275279	\$0.51
ClaimServiceFee(·)	33103	\$0.06	zkVerify(·)	2286423	\$4.20

What we have so far

1. Disincentivize unobserved unrestricted collusion in **finite** PIR services with k servers (rational or malicious), and positive service fees
2. Disincentivize unobserved unrestricted collusion in **infinite** PIR services with $\ell \gg k$ servers (rational or malicious), positive rewards and positive service fees
3. Small computation and communication overhead and general applicability

More in the paper

The protocol, adversarial exiting strategies, strong coalitions with absolute trust for members, setting the evidence collection time window Δ, non-triviality of functions, blind collusion, etc.

Future directions

1. Advance the current analysis
   A. More practical solutions for large γ in γ-nontrivial information gain
2. Derive solutions for other protocols with the non-collusion assumption
   A. Robust PIR where not all responses are needed for reconstruction
   B. Other protocols that employ this assumption, including generic or outsourced **multi-party computation** (MPC*), **secret sharing schemes** ([Working paper]), **distributed key generation**, **time-release encryption**, etc.
   (*The current approach generalizes to 2PC and MPC in dishonest majority setting.)

Thank you.