Co-designing Cryptographic Systems with Resource-Constrained Hardware

Jean-Luc Watson

Dissertation Talk
5/7/24

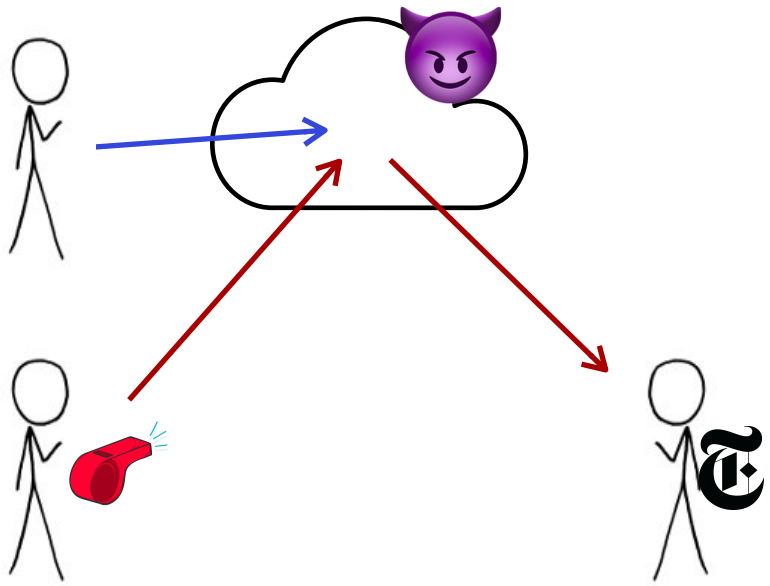
New cryptographic primitives are very powerful

New cryptographic primitives are very powerful

Metadata-hiding communication

[CGB+15, KCG+17, ECG+21, AS16, …]

Disguise communication patterns so that interested observers cannot tell who is talking!

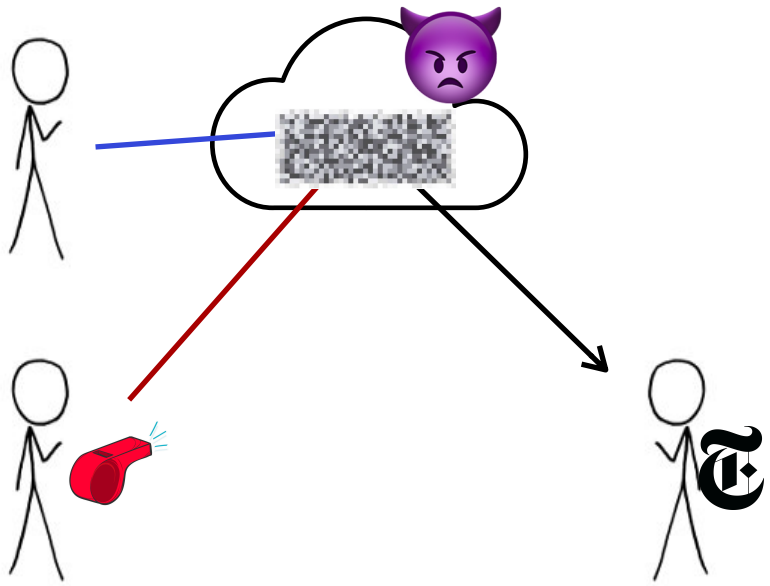


New cryptographic primitives are very powerful

Metadata-hiding communication

[CGB+15, KCG+17, ECG+21, AS16, …]

Disguise communication patterns so that interested observers cannot tell who is talking!

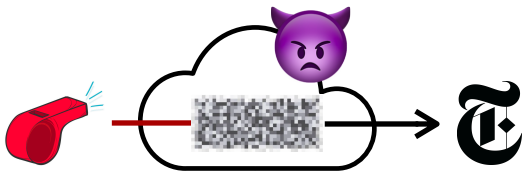


New cryptographic primitives are very powerful

Metadata-hiding communication

[CGB+15, KCG+17, ECG+21, AS16, …]

Disguise communication patterns so that interested observers cannot tell who is talking!

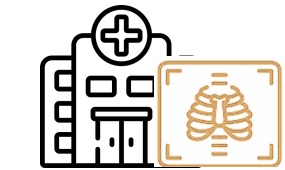


New cryptographic primitives are very powerful

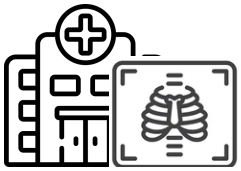
Multi-party computation

[Yao86, GMW87]

Compute a shared result from each of your private inputs without revealing them!



Hospital 1



Hospital 3

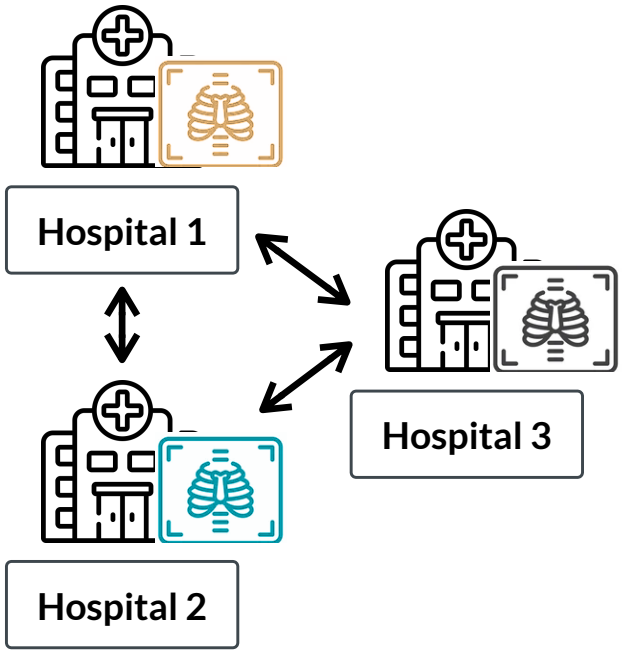
Hospital 2

New cryptographic primitives are very powerful

Multi-party computation

[Yao86, GMW87]

Compute a shared result from each of your private inputs without revealing them!

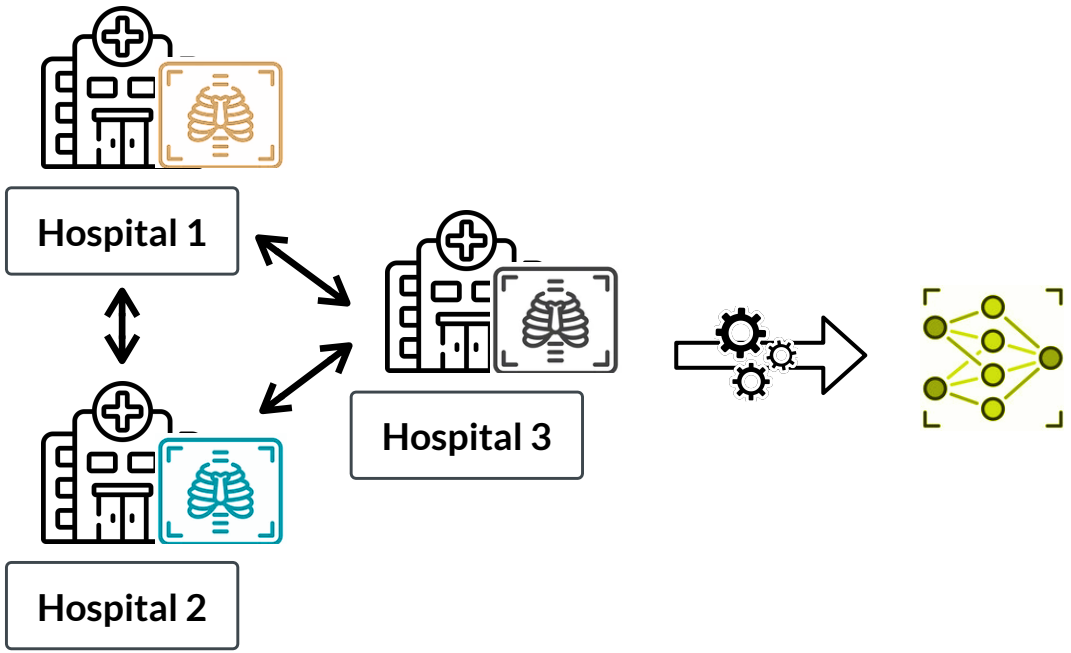


New cryptographic primitives are very powerful

Multi-party computation

[Yao86, GMW87]

Compute a shared result from each of your private inputs without revealing them!

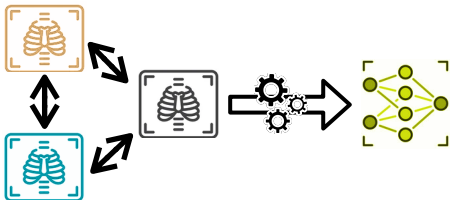


New cryptographic primitives are very powerful

Multi-party computation

[Yao86, GMW87]

Compute a shared result from each of your private inputs without revealing them!

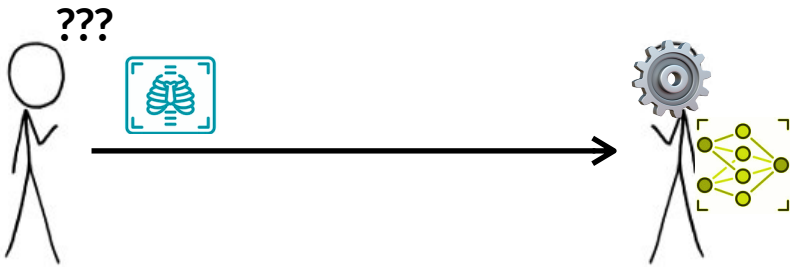


New cryptographic primitives are very powerful

Zero-knowledge Proofs

[FFS87]

Prove that you know something without revealing *what* it is!



New cryptographic primitives are very powerful

Zero-knowledge Proofs

[FFS87]

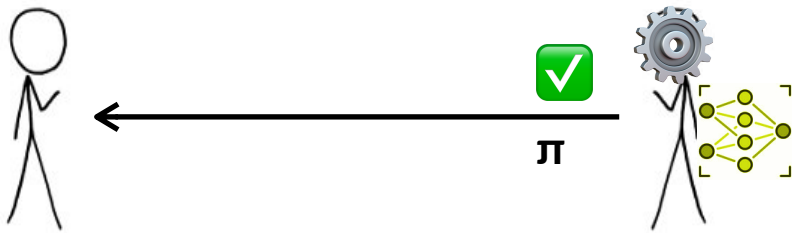
Prove that you know something without revealing *what* it is!

New cryptographic primitives are very powerful

Zero-knowledge Proofs

[FFS87]

Prove that you know something without revealing *what* it is!

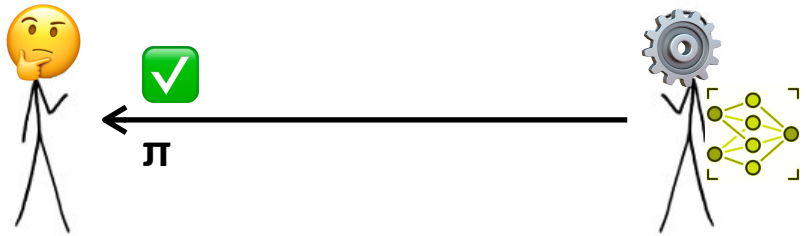


New cryptographic primitives are very powerful

Zero-knowledge Proofs

[FFS87]

Prove that you know something without revealing *what* it is!

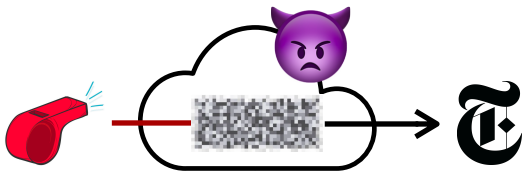


New cryptographic primitives are very powerful

Metadata-hiding communication

[CGB+15, KCG+17, ECG+21, AS16, …]

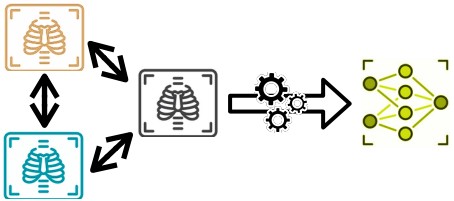
Disguise communication patterns so that interested observers cannot tell who is talking!



Multi-party computation

[Yao86, GMW87]

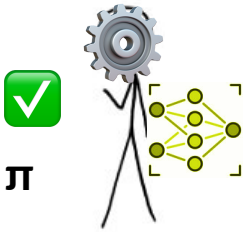
Compute a shared result from each of your private inputs without revealing them!



Zero-knowledge Proofs

[FFS87]

Prove that you know something without revealing *what* it is!



However! With great power comes great overheads

Metadata-hiding communication

[CGB+15, KCG+17, ECG+21, AS16, …]

System throughput with 10k mailboxes: **< 50 msgs/sec**

[ECG+21]

Multi-party computation

[Yao86, GMW87]

Zero-knowledge Proofs

[FFS87]

However! With great power comes great overheads

Metadata-hiding communication

[CGB+15, KCG+17, ECG+21, AS16, ...]

System throughput with 10k mailboxes: **< 50 msgs/sec** [ECG+21]

Multi-party computation

[Yao86, GMW87]

Estimated training time for VGG16: **2 weeks** [WTB+21]

Zero-knowledge Proofs

[FFS87]

However! With great power comes great overheads

Metadata-hiding communication

[CGB+15, KCG+17, ECG+21, AS16, ...]

System throughput with 10k mailboxes: **< 50 msgs/sec** [ECG+21]

Multi-party computation

[Yao86, GMW87]

Estimated training time for VGG16: **2 weeks** [WTB+21]

Zero-knowledge Proofs

[FFS87]

Proving a correct ECDSA signature: **45 sec***

*https://github.com/0xPARC/circom-ecdsa#benchmarks

However! With great power comes great overheads

Metadata-hiding communication

[CGB+15, KCG+17, ECG+21, AS16, ...]

System throughput with 10k mailboxes: **< 50 msgs/sec** [ECG+21]

Multi-party computation

[Yao86, GMW87]

Estimated training time for VGG16: **2 weeks** [WTB+21]

Zero-knowledge Proofs

[FFS87]

Proving a correct ECDSA signature: **45 sec***

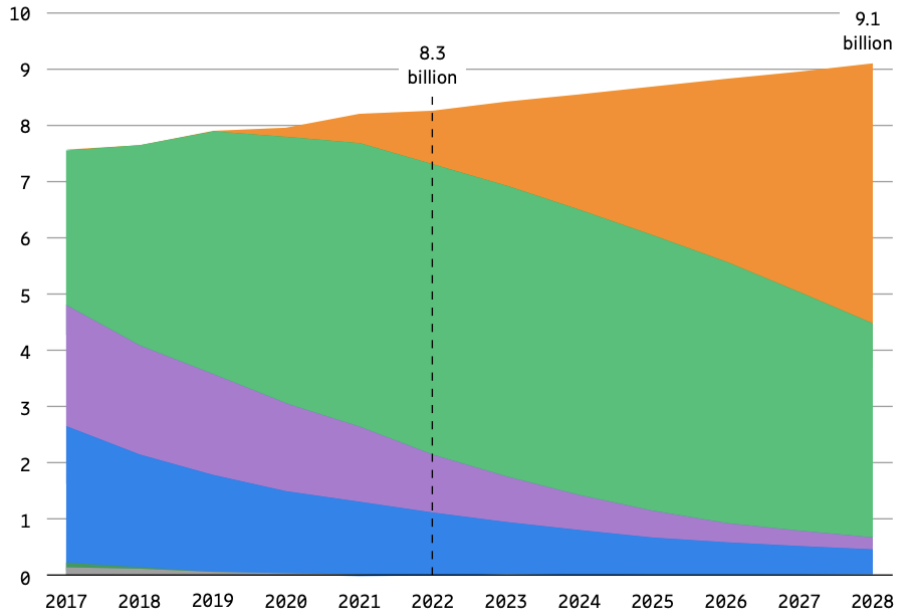
Let's take advantage of new heterogeneous hardware!

*https://github.com/0xPARC/circom-ecdsa#benchmarks

There's **more** hardware out there!

Billions of mobile phone customers with a processor in their hand

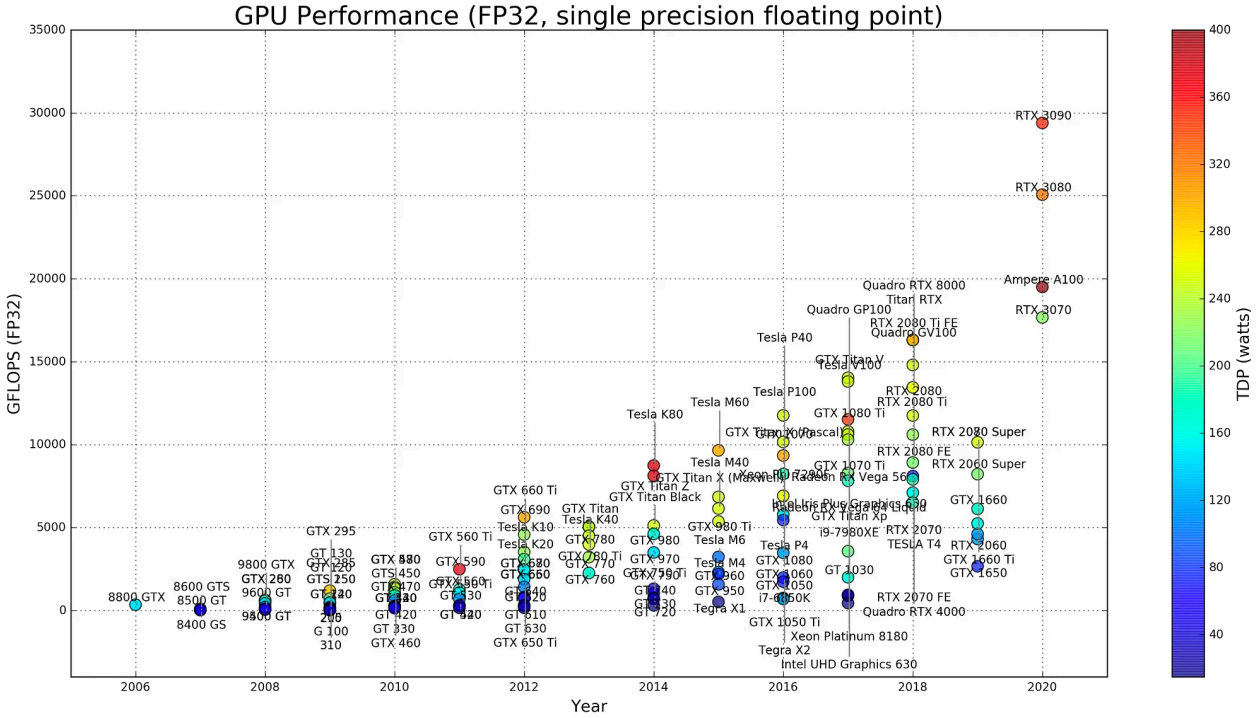
Figure 1: Mobile subscriptions by technology (billion)



https://www.ericsson.com/49dd9d/assets/local/reports-papers/mobility-report/documents/2023/ericsson-mobility-report-june-2023.pdf

There's **faster** hardware out there!

Increasingly-powerful GPUs are commonplace



https://blog.inten.to/hardware-for-deep-learning-part-3-gpu-8906c1644664

Naturally, platforms have some fundamental constraints

- Mobile platforms (e.g. embedded sensors, mobile phones)

More compute devices BUT limited on-device power and processing speed

Naturally, platforms have some fundamental constraints

- Mobile platforms (e.g. embedded sensors, mobile phones)

  More compute devices BUT limited on-device power and processing speed

- Hardware accelerators (e.g. GPUs)

  More compute cores BUT limited on-device memory

Naturally, platforms have some fundamental constraints

- Mobile platforms (e.g. embedded sensors, mobile phones)

  More compute devices BUT limited on-device power and processing speed

- Hardware accelerators (e.g. GPUs)

  More compute cores BUT limited on-device memory

My work: how to build cryptographic systems with these tradeoffs

Thesis

Real-world applications can feasibly leverage advanced cryptographic primitives without completely sacrificing performance by restructuring the primitives to move protocol execution to resource-constrained devices.

Thesis

Problems at a useful scale

Real-world applications can feasibly leverage advanced cryptographic primitives without completely sacrificing performance by restructuring the primitives to move protocol execution to resource-constrained devices.

Thesis

No free lunch, but an affordable one

Real-world applications can **feasibly leverage advanced cryptographic primitives without completely sacrificing performance** by restructuring the primitives to move protocol execution to resource-constrained devices.

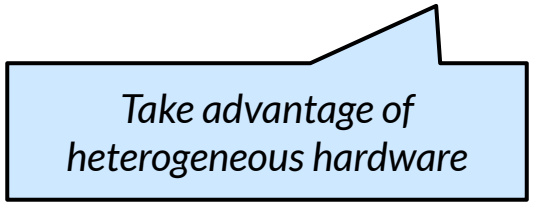
Thesis

Real-world applications can feasibly leverage advanced cryptographic primitives without completely sacrificing performance by **restructuring the primitives** to move protocol execution to resource-constrained devices.

Systems-level modifications around existing primitives

Thesis

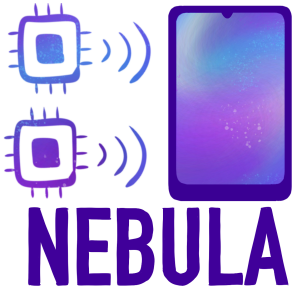
Real-world applications can feasibly leverage advanced cryptographic primitives without completely sacrificing performance by restructuring the primitives to **move protocol execution to resource-constrained devices.**



This talk

Metadata-hiding communication

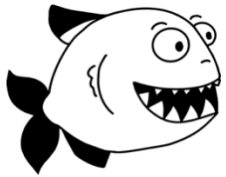
[CGB+15, KCG+17, ECG+21, AS16, …]



Mobile phones

Multi-party computation

[Yao86, GMW87]



Piranha

GPUs

Zero-knowledge Proofs

[FFS87]

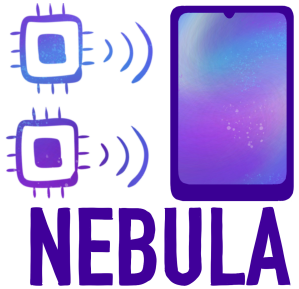
ZK Proofs at Scale

GPUs

This talk

Metadata-hiding communication

[CGB+15, KCG+17, ECG+21, AS16, …]



Mobile phones

Multi-party computation

[Yao86, GMW87]

Piranha

GPUs

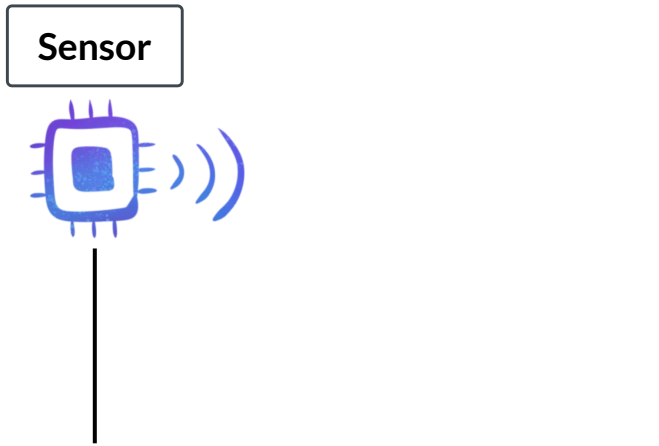
Zero-knowledge Proofs

[FFS87]

ZK Proofs at Scale

GPUs

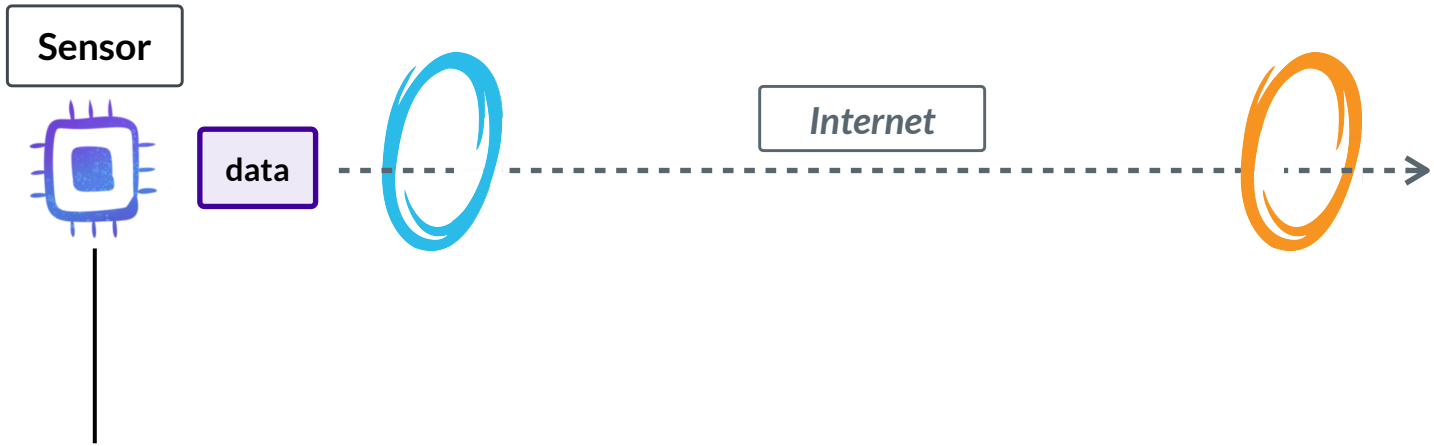
An ideal backhaul system



Monterey
Berkeley
Paris
Under a bridge in Minnesota
Fedex shipping depot

[img src]

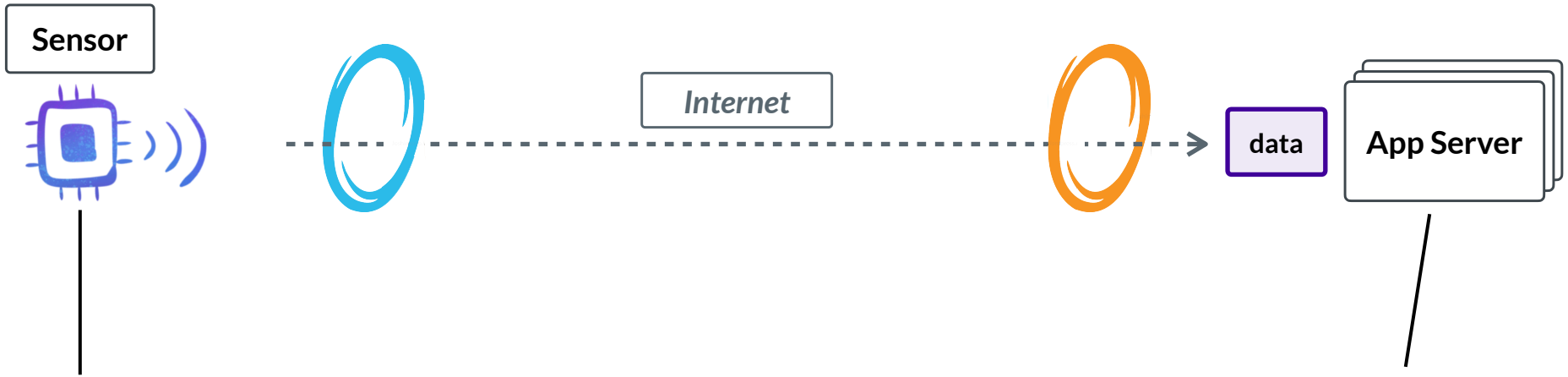
An ideal backhaul system



Monterey
Berkeley
Paris
Under a bridge in Minnesota
Fedex shipping depot

[img src]

An ideal backhaul system

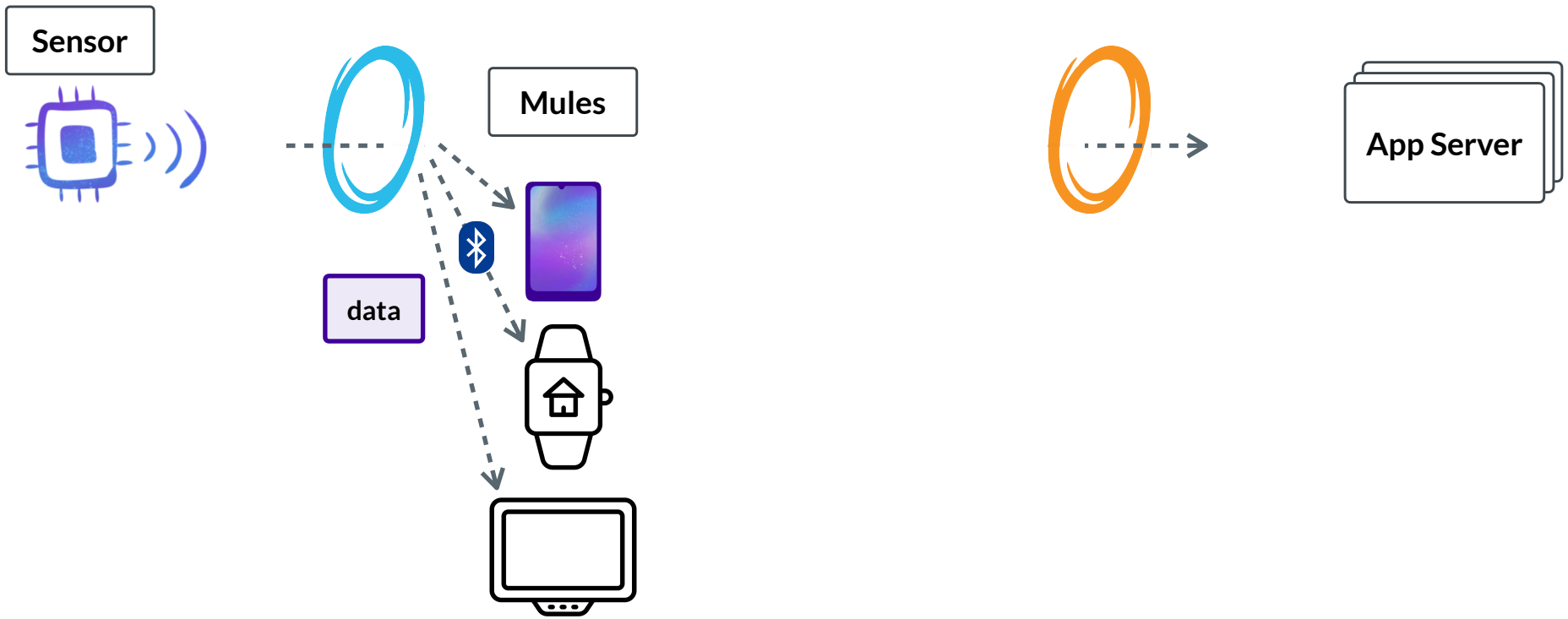


Monterey
Berkeley
Paris
Under a bridge in Minnesota
Fedex shipping depot

Airborne salinity study
Trash can servicing
Traffic conditions
Wildlife counting
Package tracking

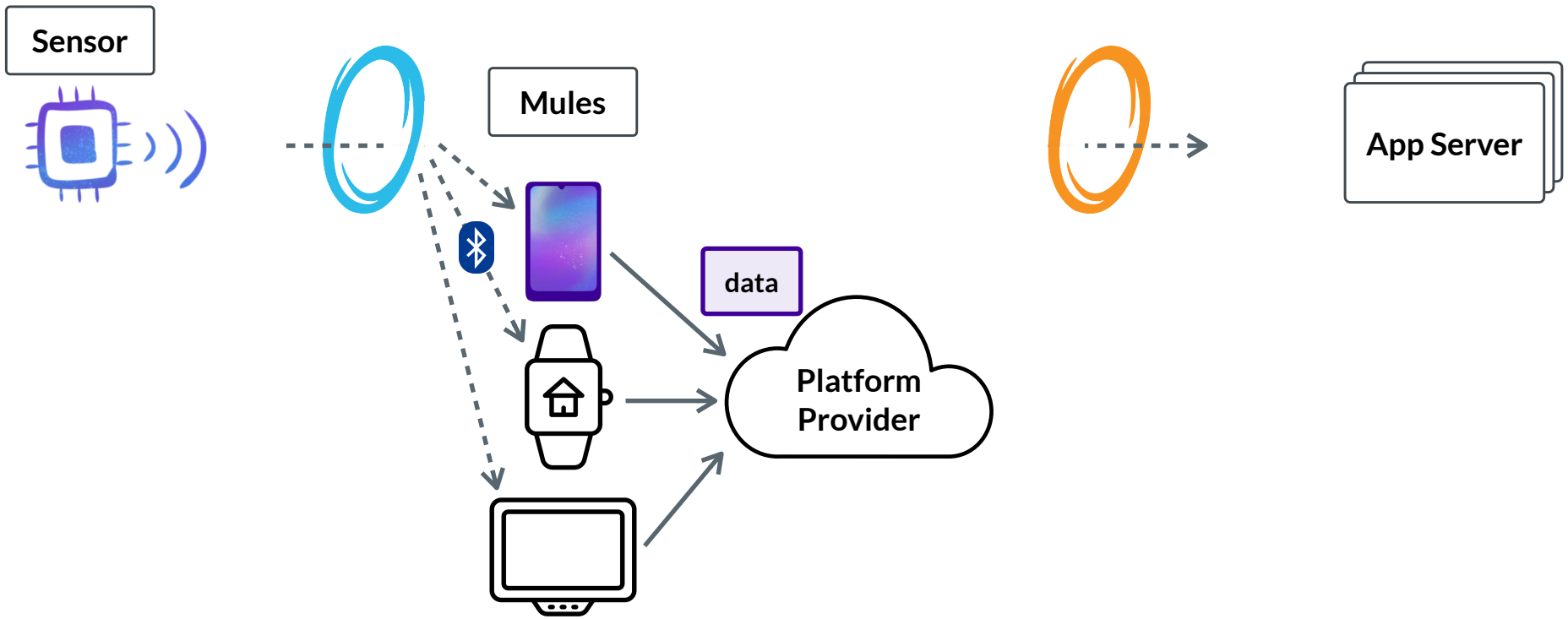
[img src]

Data is carried by 'mules' in close proximity



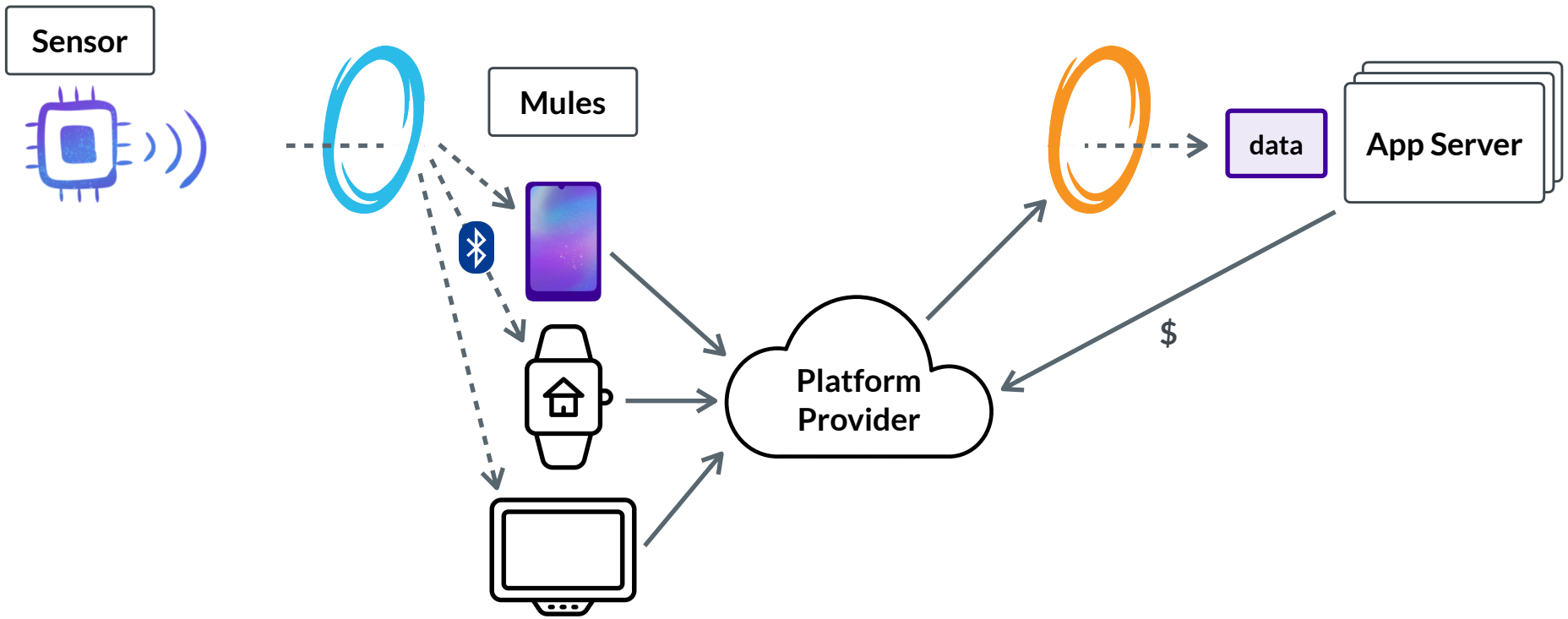
[icons by Icons Field from Noun Project]

Mules send data to a backhaul provider



[icons by Icons Field from Noun Project]

The provider forwards to paying servers



[icons by Icons Field from Noun Project]

Mules send data to a backhaul provider

Sensor

Backhaul systems are *already* being deployed

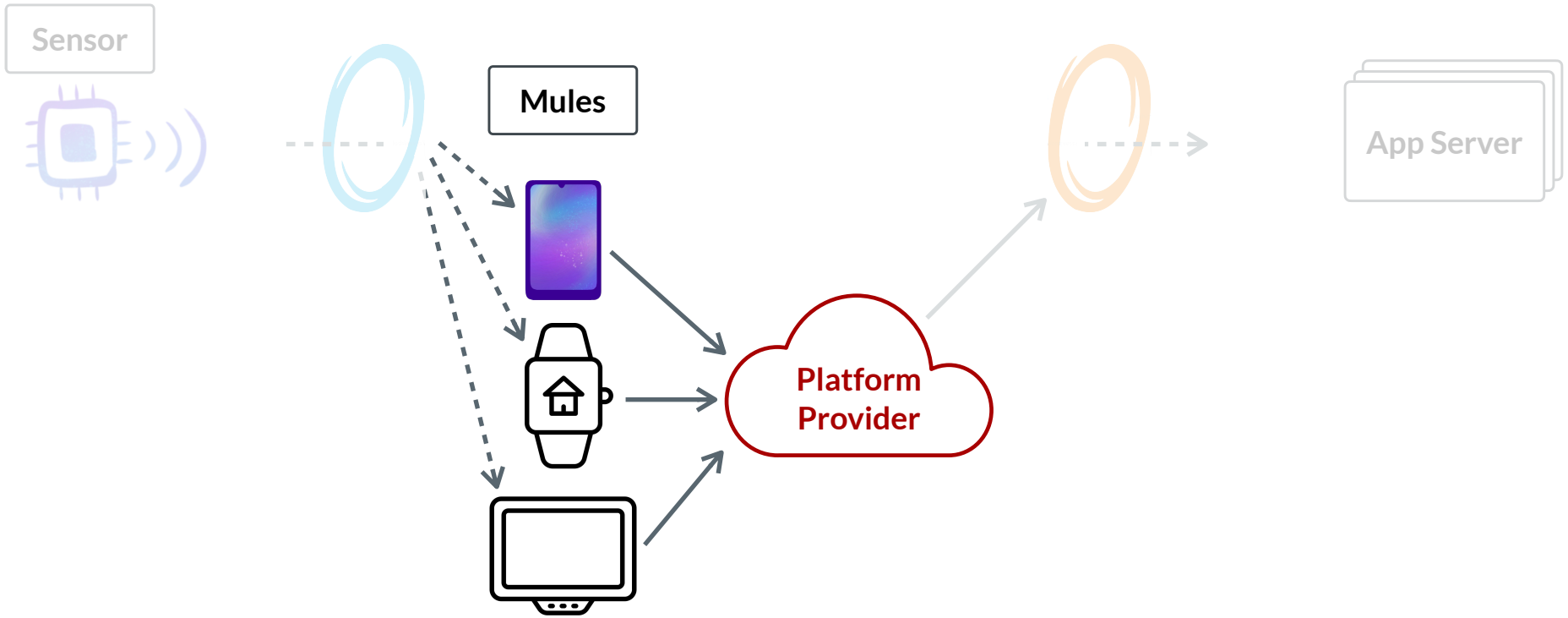
Find My

p Server

[icons by Icons Field from Noun Project]

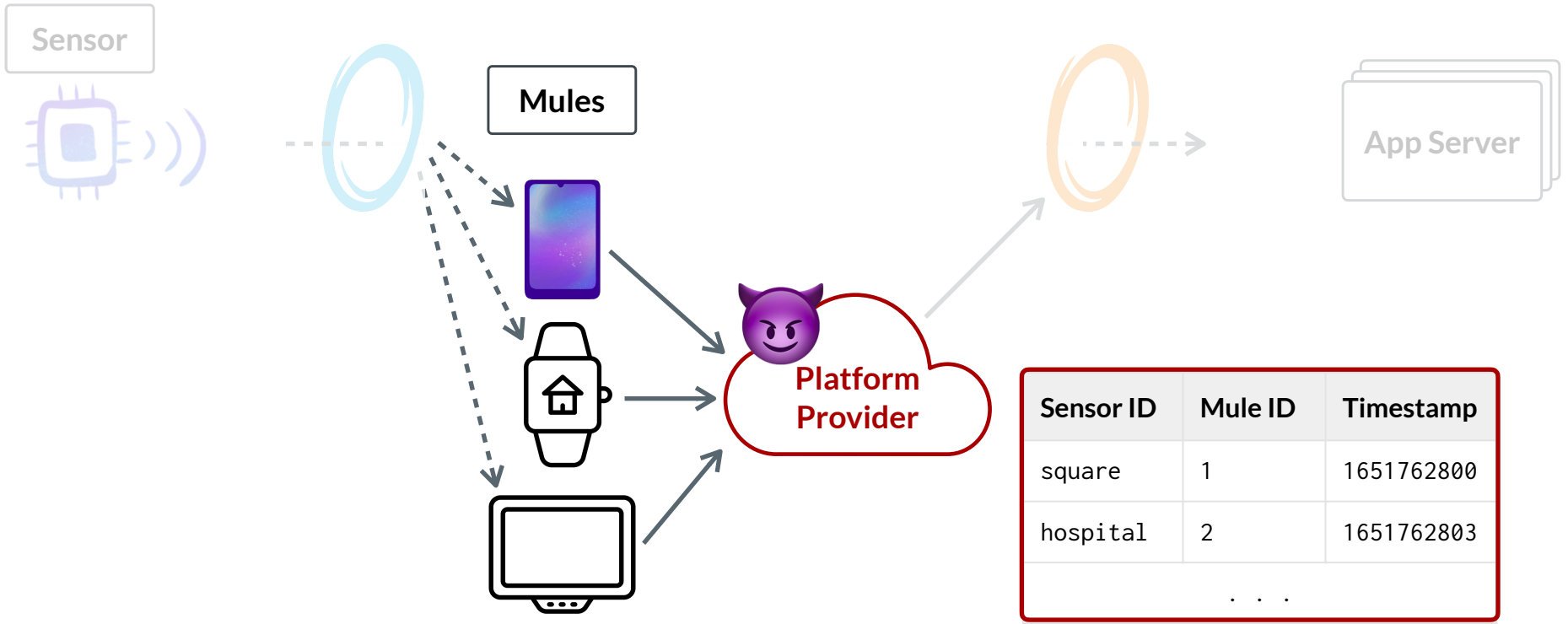
[img src]

Centralized backhaul threatens privacy



[icons by Icons Field from Noun Project]

Centralized backhaul threatens privacy

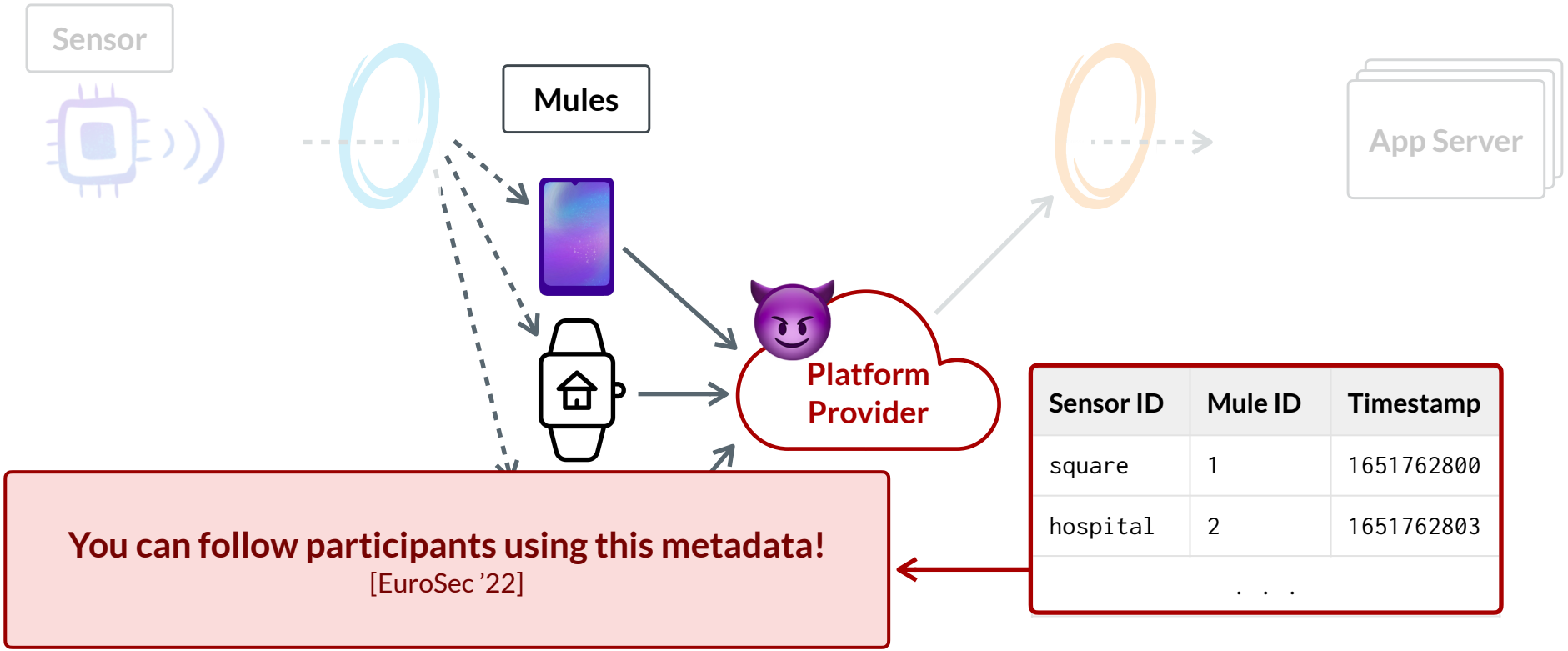


Sensor ID	Mule ID	Timestamp
square	1	1651762800
hospital	2	1651762803
. . .		

[icons by Icons Field from Noun Project]

[img src]

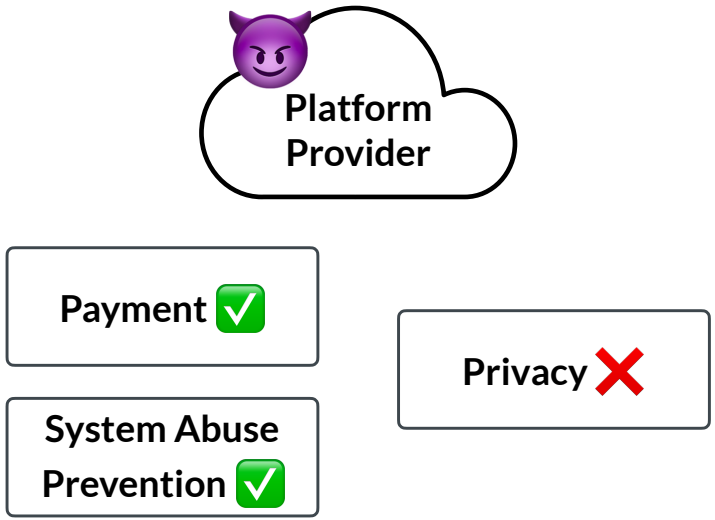
Centralized backhaul threatens privacy



[icons by Icons Field from Noun Project]

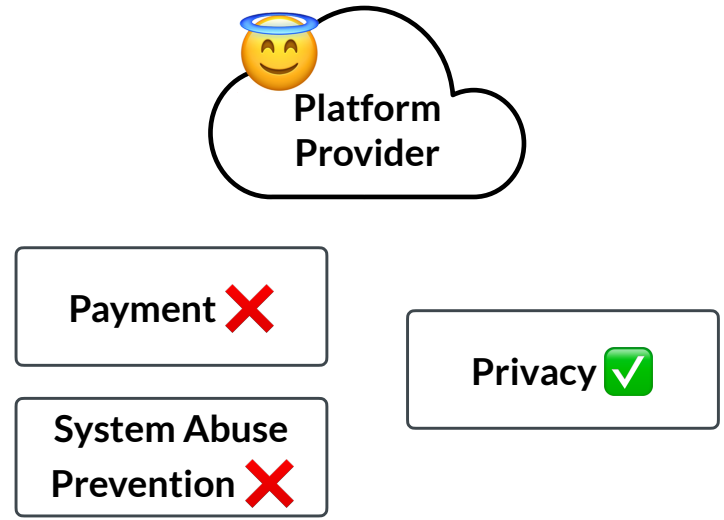
Problem

Provider can run a valuable service if it is not concerned about privacy



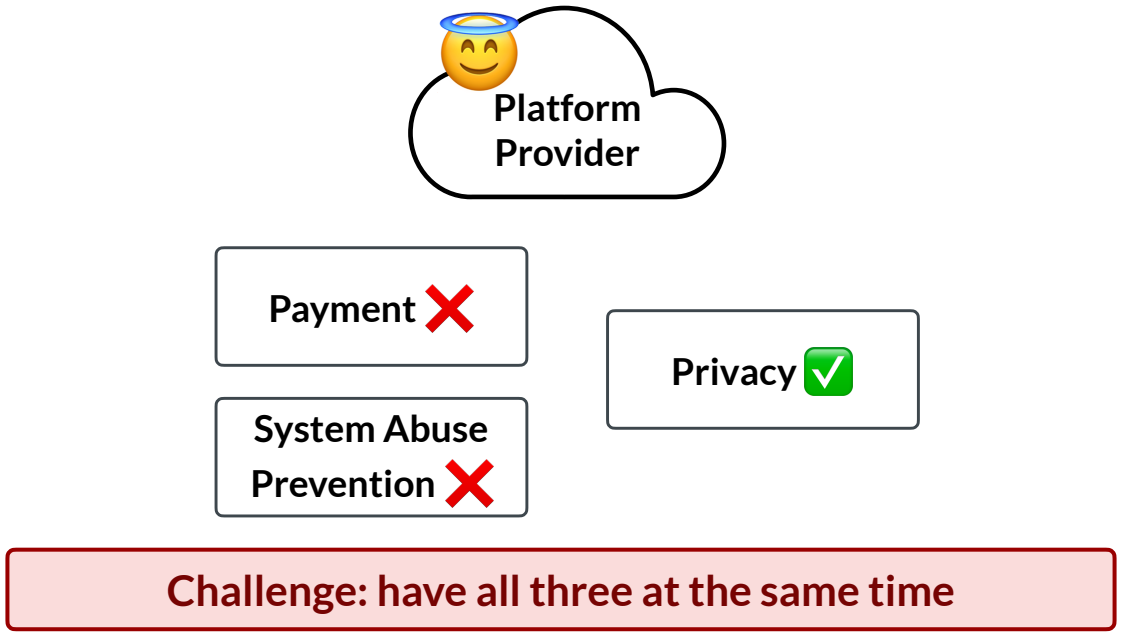
Problem

A provider can trivially support privacy by allowing any behavior

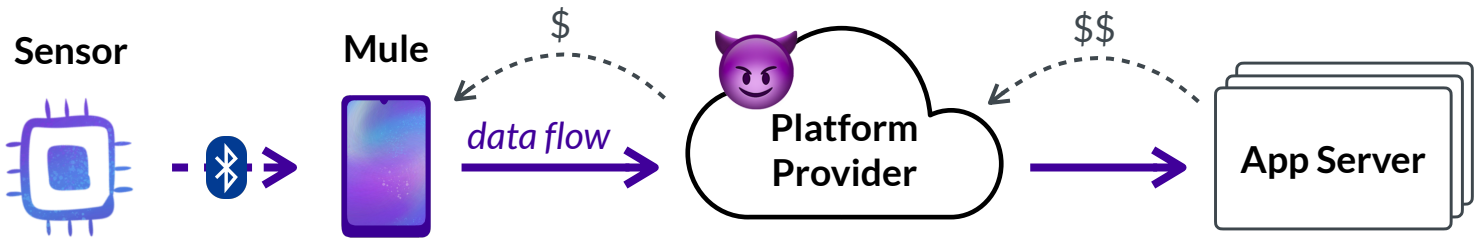


Problem

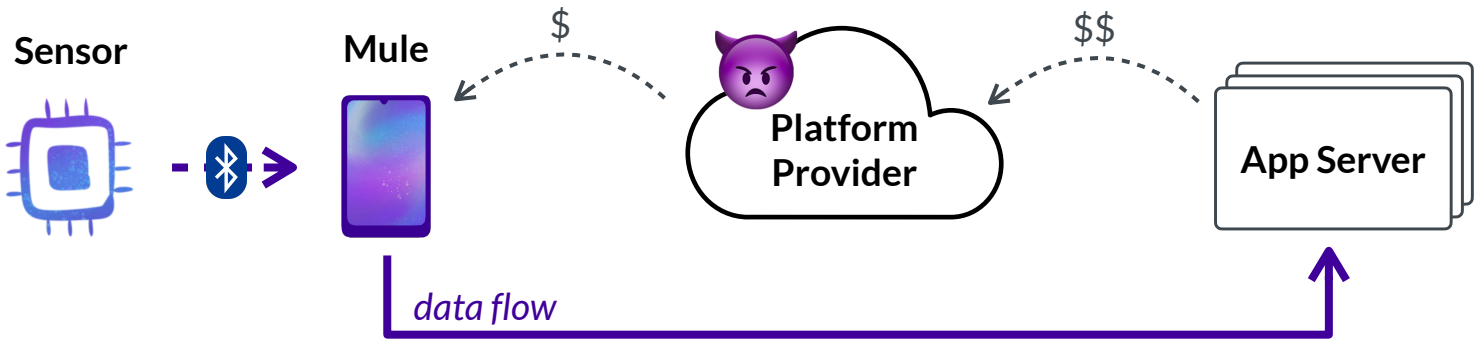
A provider can trivially support privacy by allowing any behavior



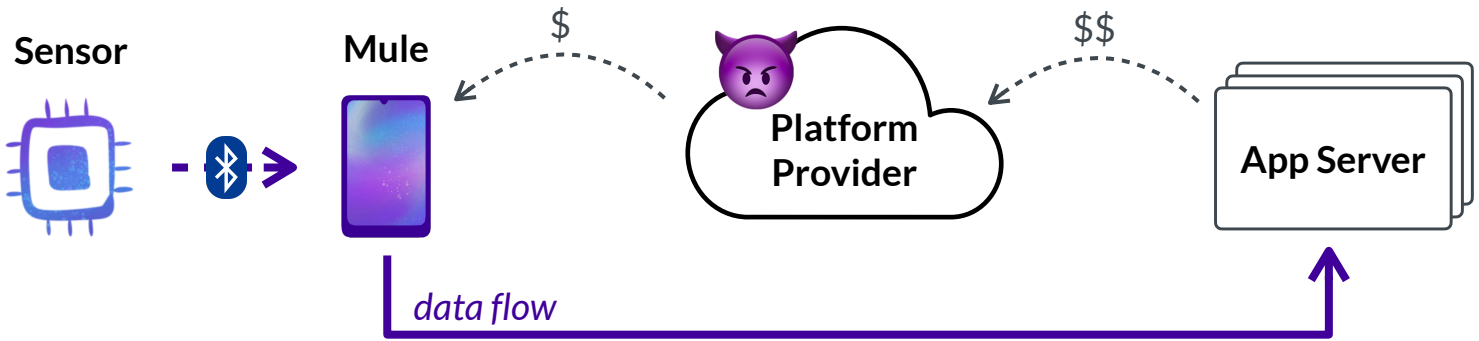
Nebula’s approach: decentralize protocol onto mules



Nebula’s approach: decentralize protocol onto mules



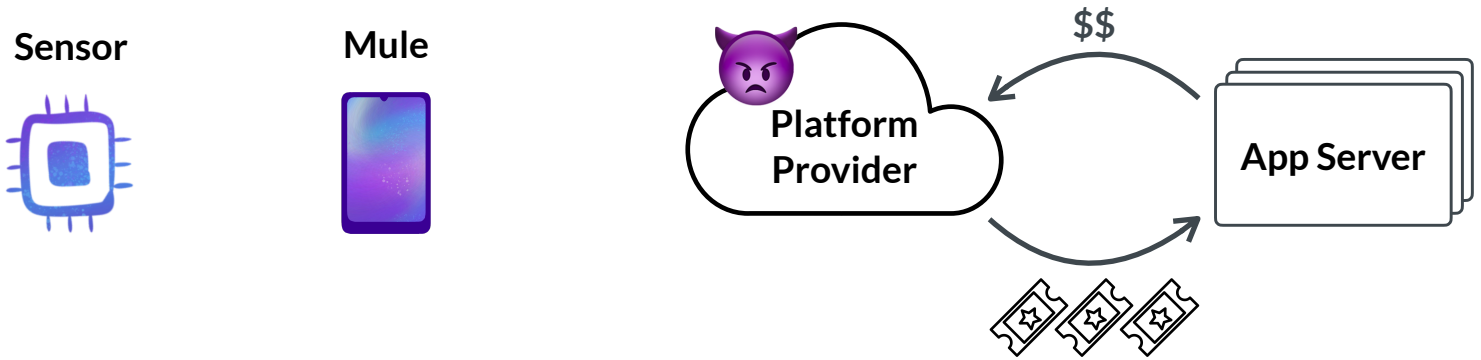
Nebula's approach: decentralize protocol onto mules



Out-of-band accounting: app servers buy untraceable *tokens* from provider and exchange them for data.

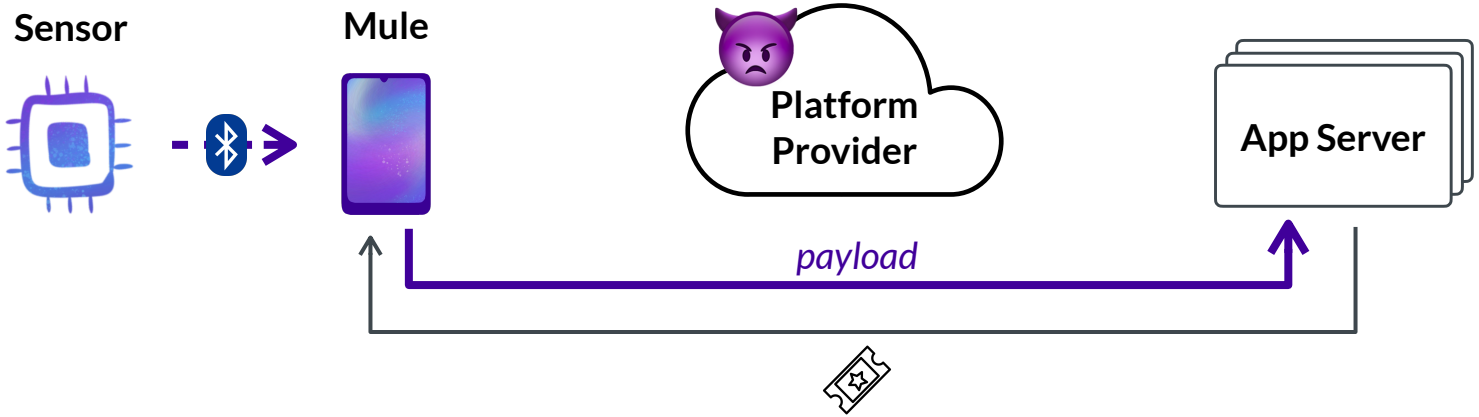
(1) Token Pre-purchase

- At the beginning of an epoch, app servers *pre-purchase* tokens from the provider



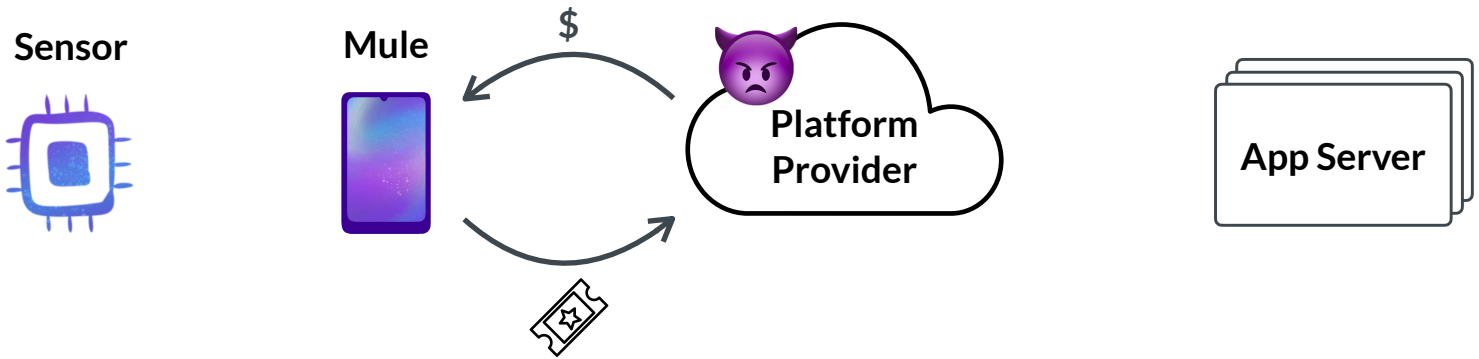
(2) Payload Delivery

- Mules send data directly to application servers over anonymous connections and receive tokens in exchange

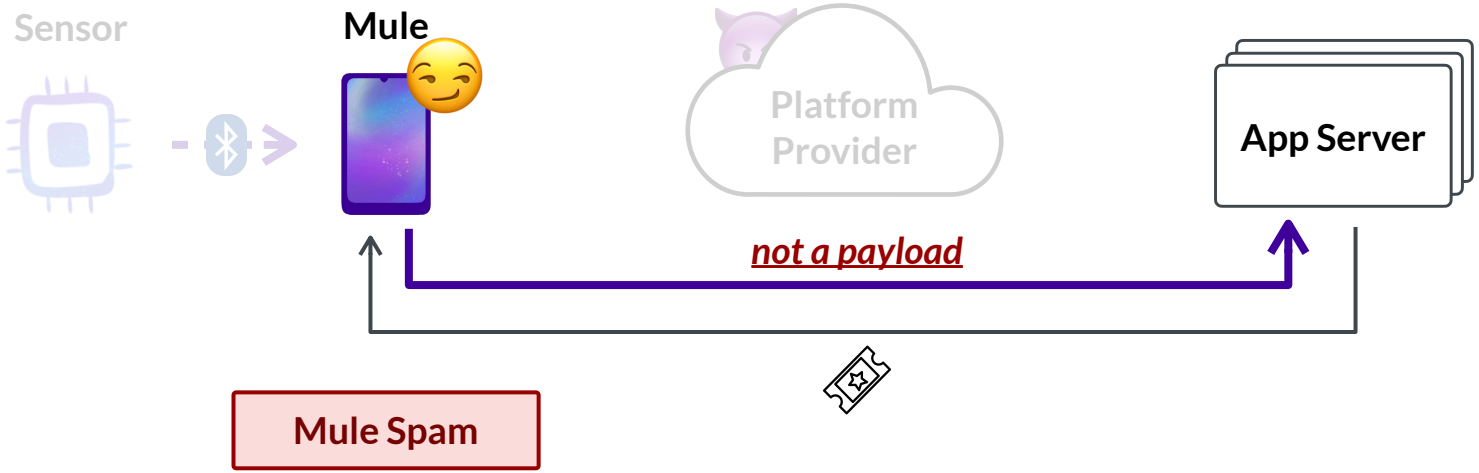


(3) Token Redemption

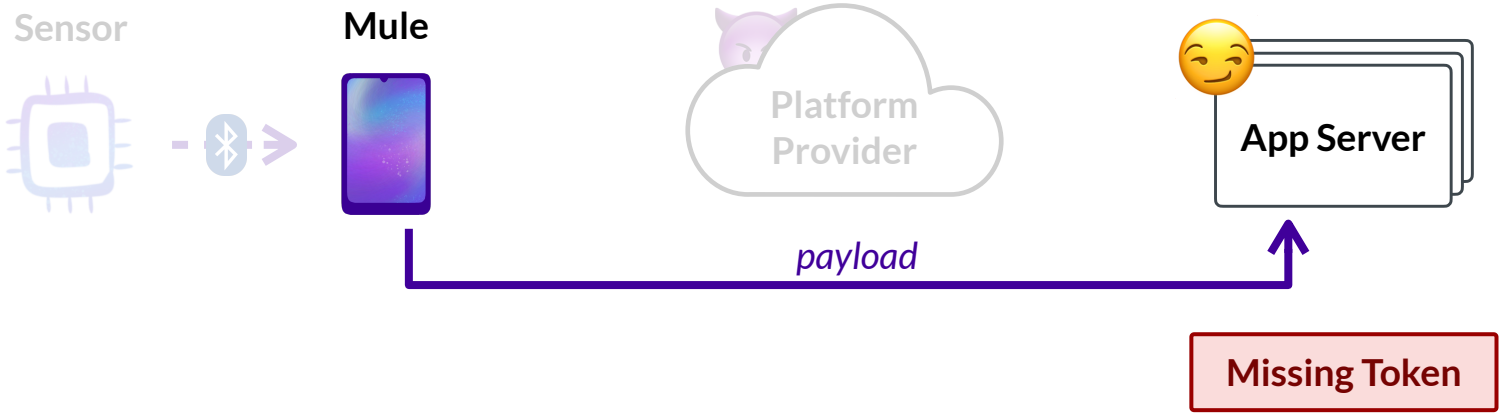
- At the end of an epoch, mules redeem tokens with the platform provider in exchange for compensation



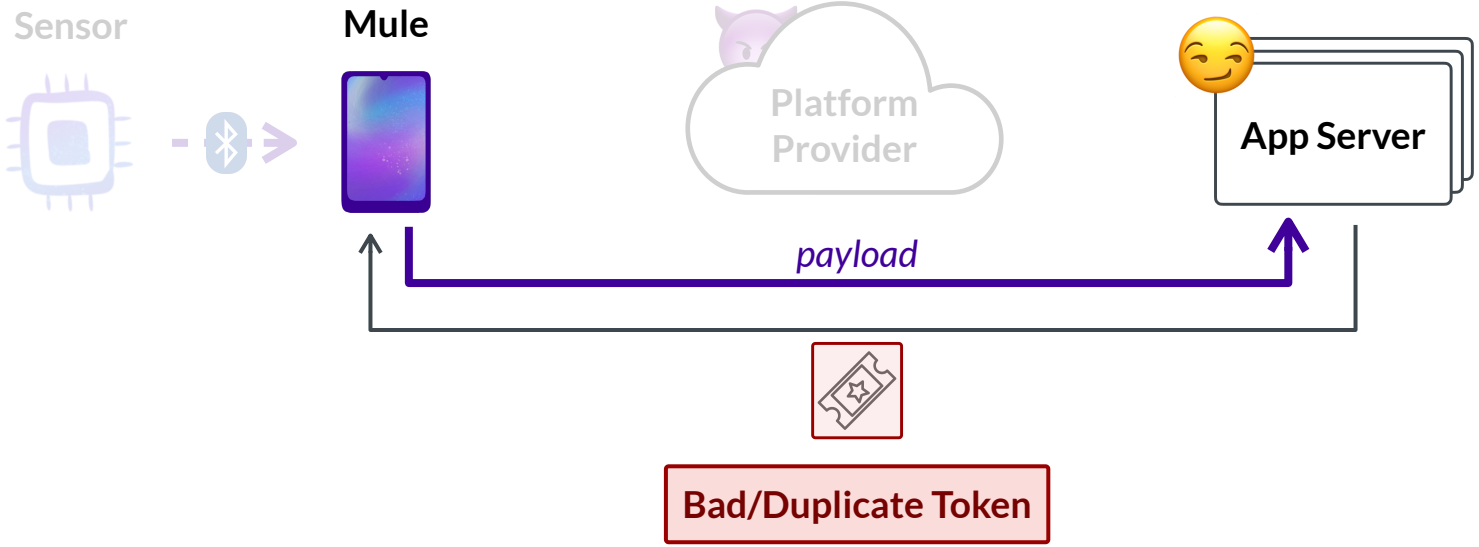
Big problem: **delivery misbehavior**



Big problem: **delivery misbehavior**

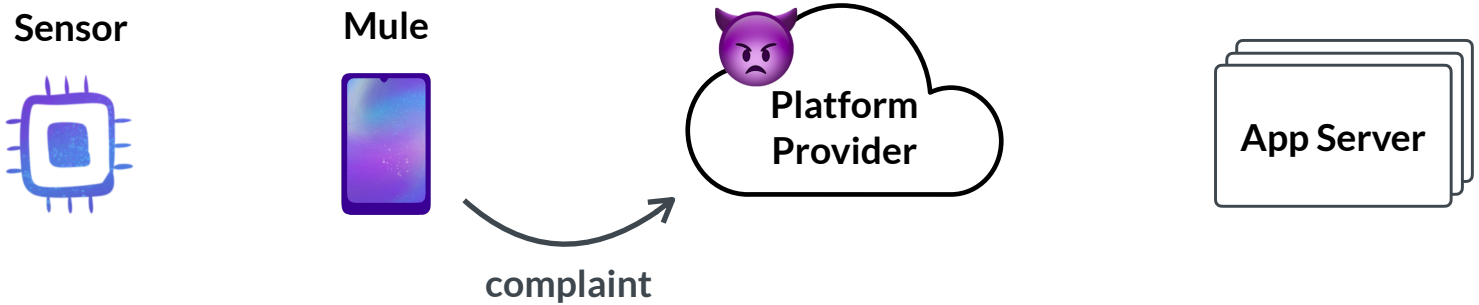


Big problem: **delivery misbehavior**



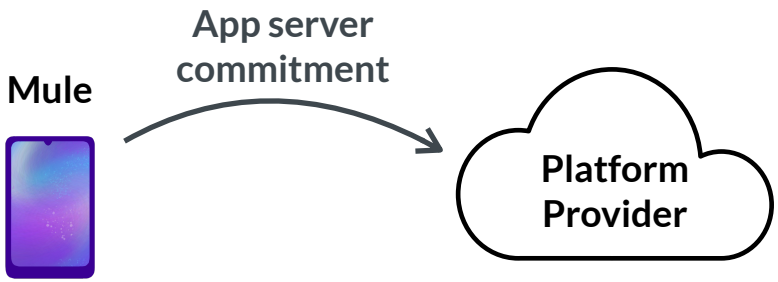
(4) Complaint

- After an epoch, if a mule notices misbehavior (e.g. invalid token), it can *complain* to the provider for a new one



Complaints are based on app server commitments

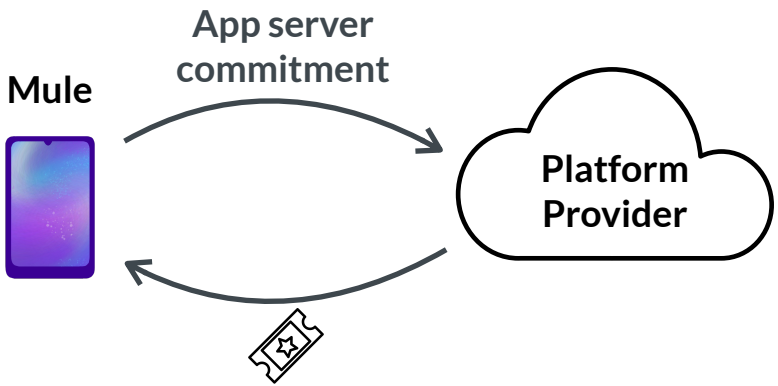
Before receiving data, app servers commit to token they will use if payload is uploaded



Complaints are based on app server commitments

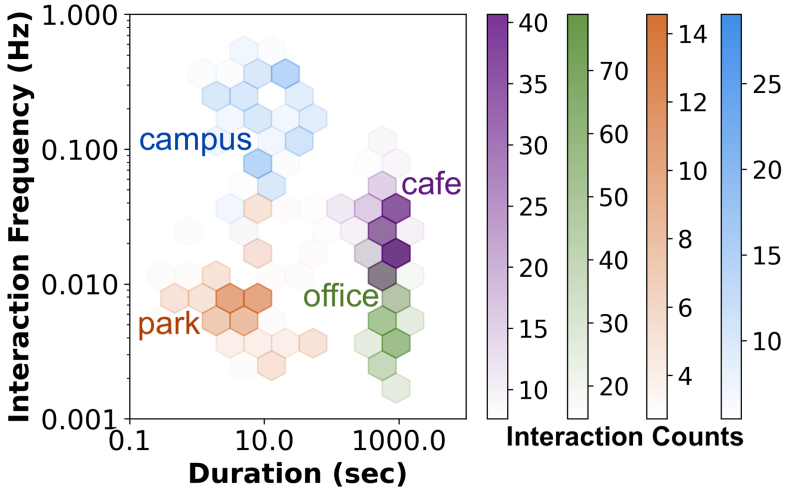
Before receiving data, app servers commit to token they will use if payload is uploaded

A *new* token is granted for *proof of misbehavior*



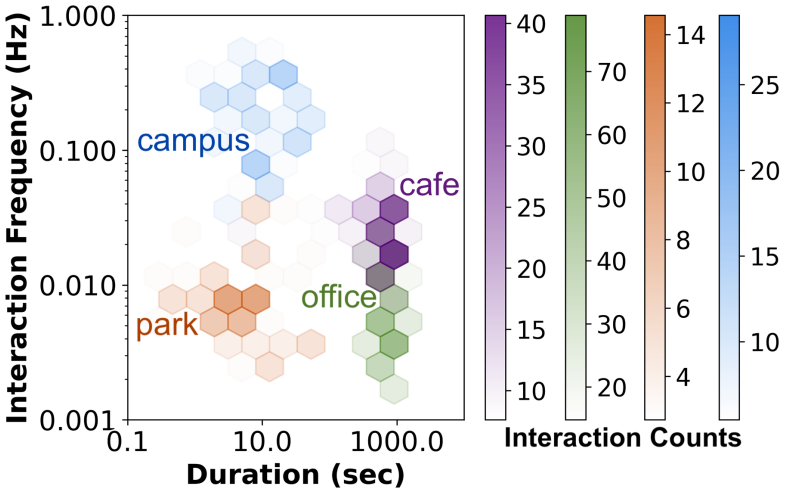
Supporting upload in varied environments

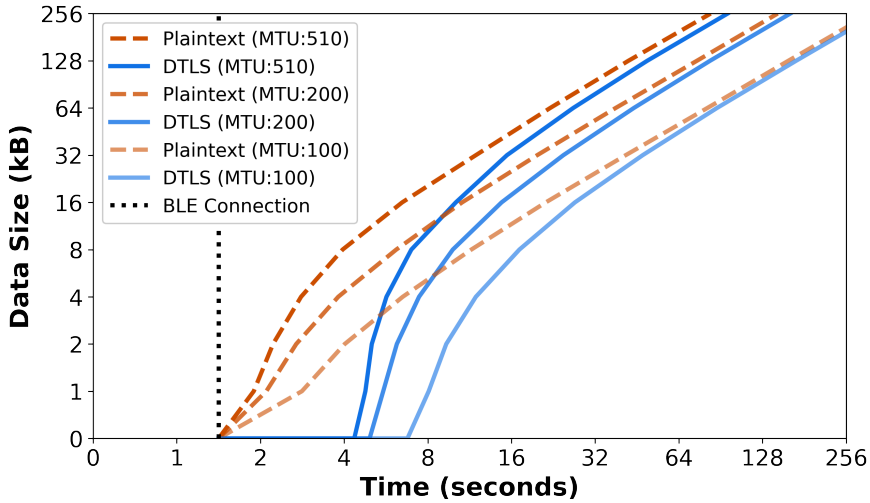
We measured likely *frequency* of sensor-mule interaction



Supporting upload in varied environments

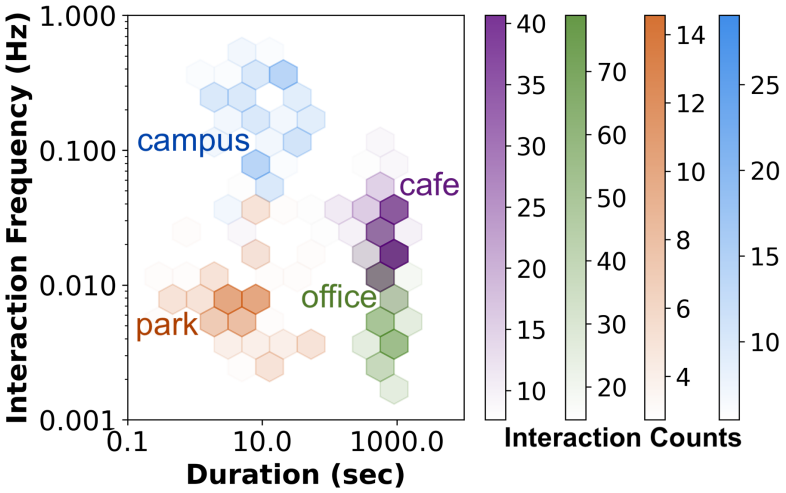
We measured likely *frequency* of sensor-mule interaction, and *duration* of data transfer

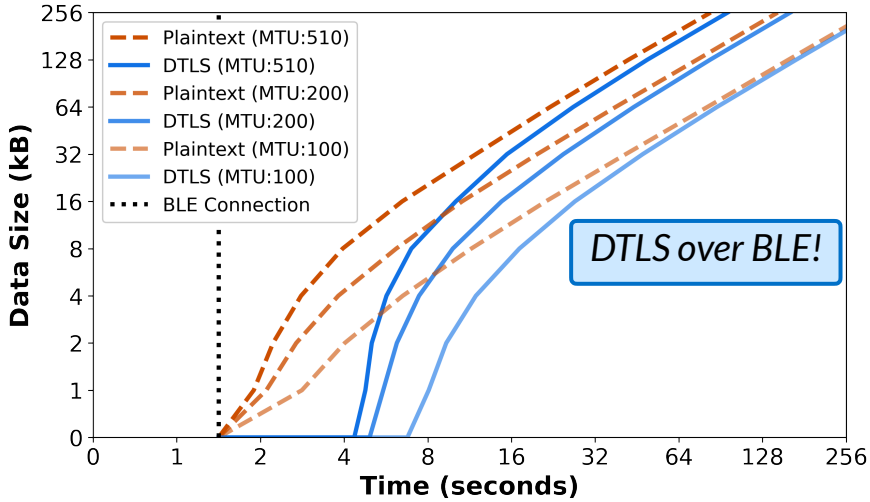




Supporting upload in varied environments

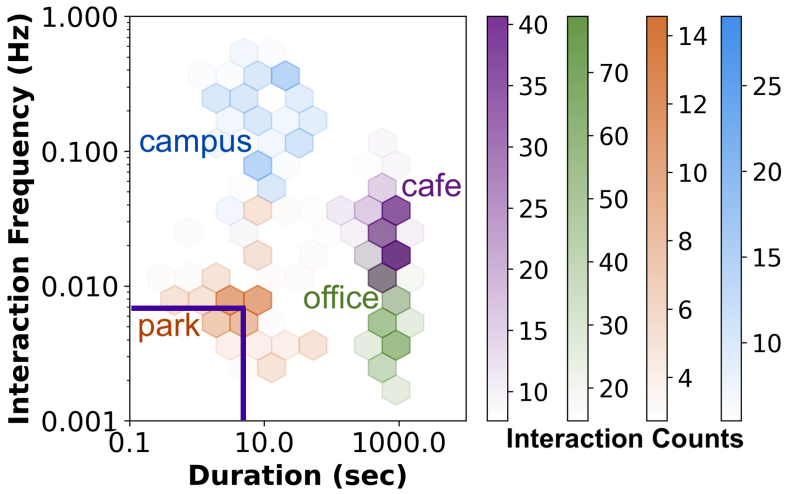
We measured likely *frequency* of sensor-mule interaction, and *duration* of data transfer

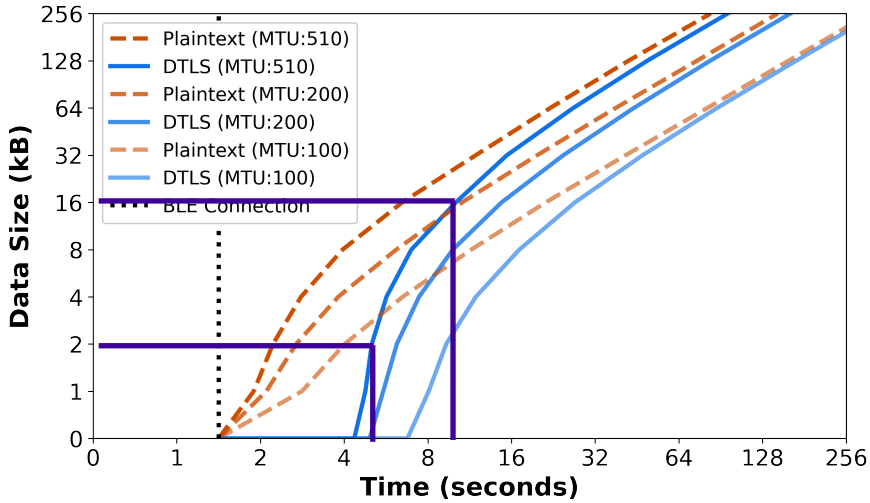




Supporting upload in varied environments

We measured likely *frequency* of sensor-mule interaction, and *duration* of data transfer

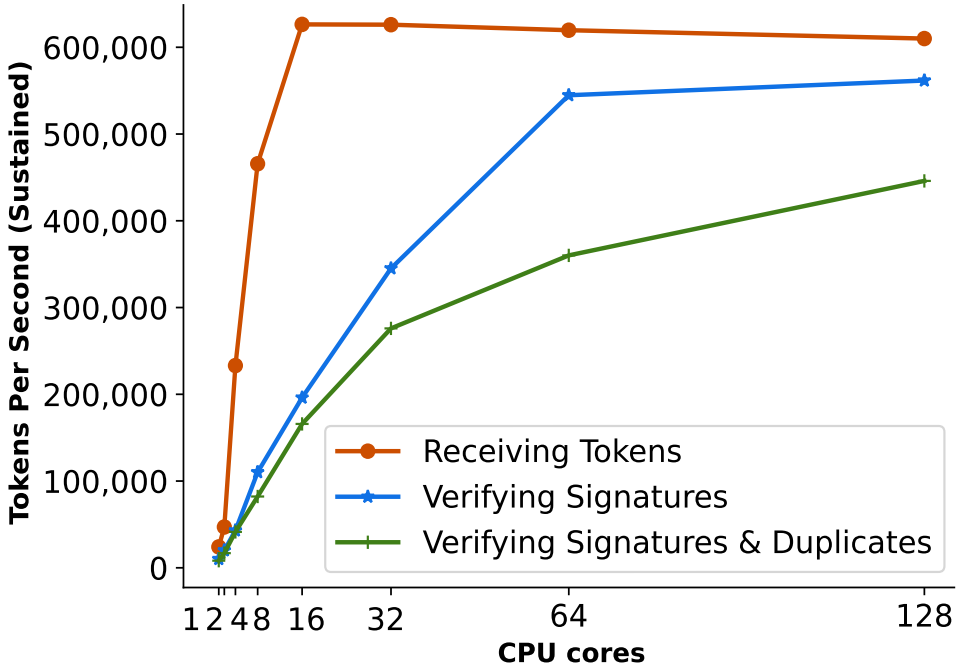




Example: in a park, we can expect a mule every few minutes,
each with a transmission window of 5 to 10 seconds, supporting 2 to 16kB payloads

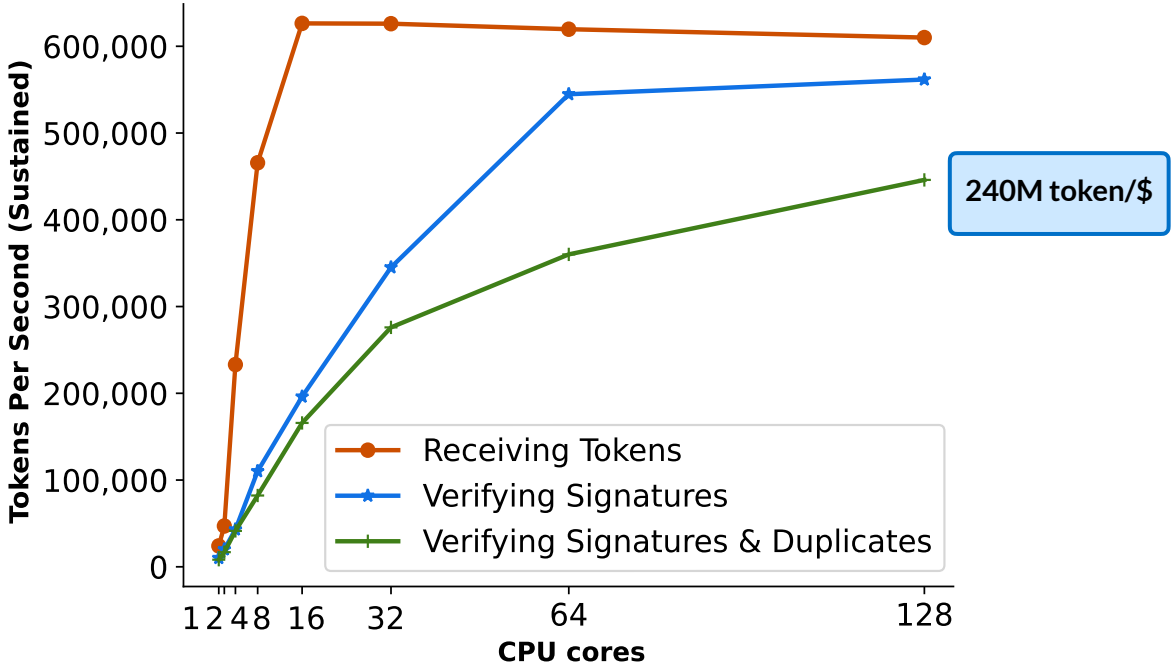
Provider performance

High-throughput database to check token validity and detect duplicates.



Provider performance

High-throughput database to check token validity and detect duplicates.



Fast enough for every person in the US to redeem tokens once a month

Summary

Backhaul networks can feasibly leverage Metadata-hiding without completely sacrificing performance by restructuring payment/abuse prevention to move protocol execution to resource-constrained phones.

This talk

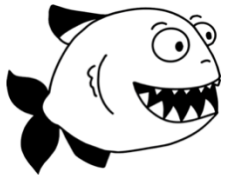
Metadata-hiding communication

[CGB+15, KCG+17, ECG+21, AS16, ...]

Mobile phones

Multi-party computation

[Yao86, GMW87]



Piranha

GPUs

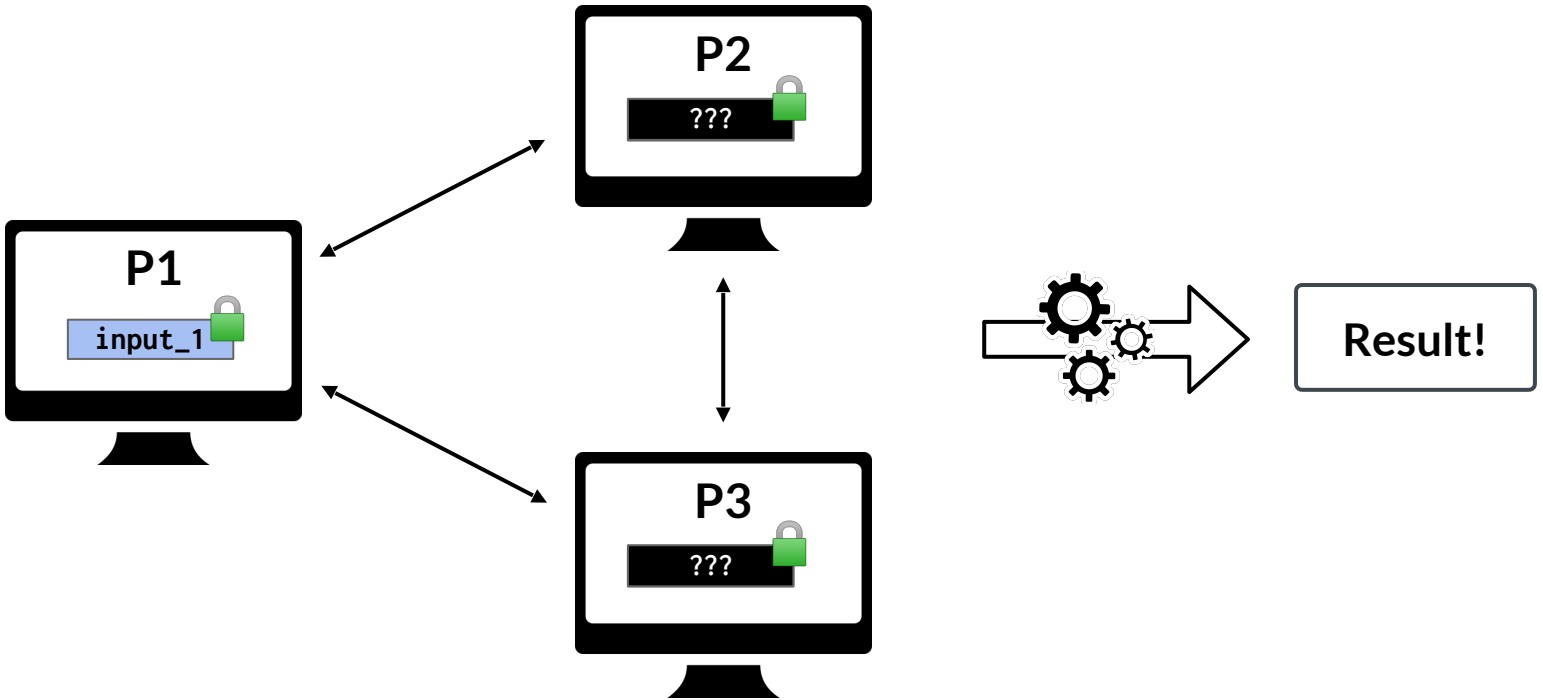
Zero-knowledge Proofs

[FFS87]

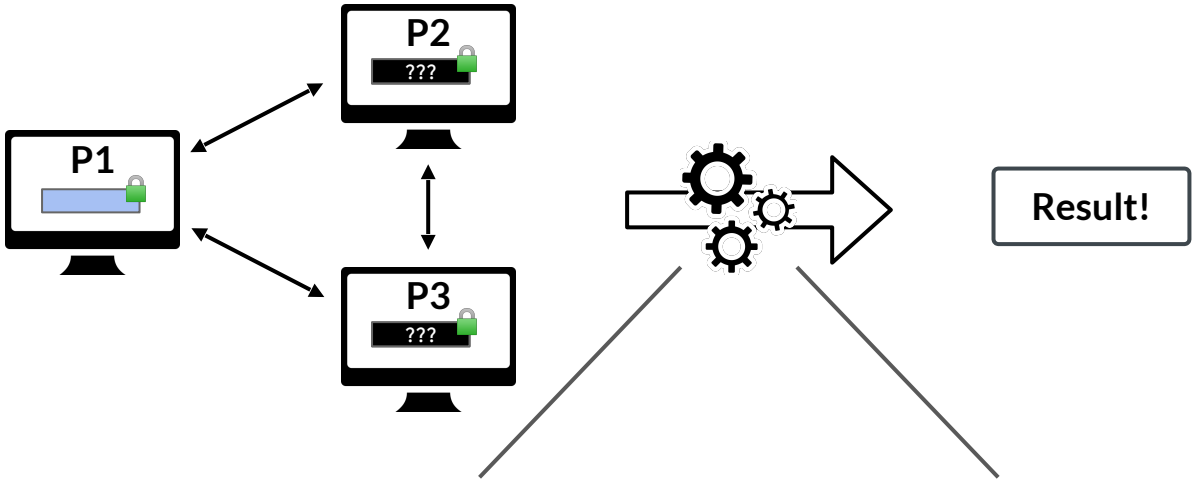
ZK Proofs at Scale

GPUs

Secure multi-party computation (MPC) [Yao86, GMW87]

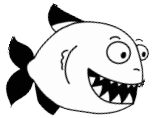


MPC has a performance problem

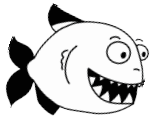


	Plaintext	MPC-based
AES Encryption	< 100 ns[1]	***~1 ms / block*** [DG21]
ML Inference (VGG16)	58 ms	***100 seconds*** [WTB+21]
ML Training (VGG16)	250 seconds	***Estimated 14 days*** [WTB+21]

[1] https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf, assuming a 3.0GHz processor

Piranha 

Goal: make accelerating secure MPC *practical*

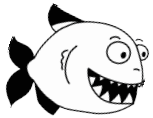
Piranha 

Goal: make accelerating secure MPC *practical*

(linear secret-sharing protocols)

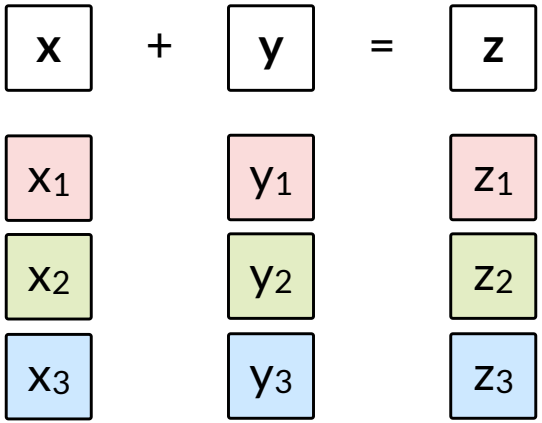
x + **y** = **z**

Piranha

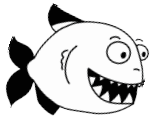


Goal: make accelerating secure MPC *practical*

(linear secret-sharing protocols)



Piranha



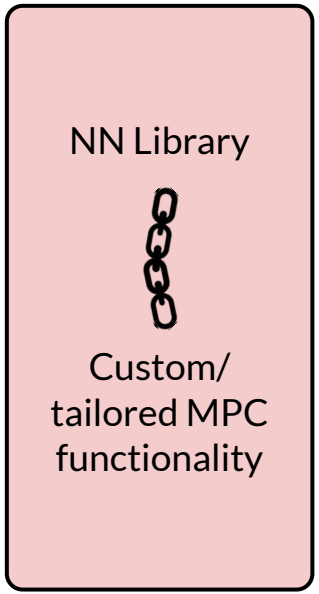
Goal: make accelerating secure MPC *practical*

(linear secret-sharing protocols)

$$x + y = z$$

$$x_1 + y_1 = z_1$$
$$x_2 + y_2 = z_2$$
$$x_3 + y_3 = z_3$$

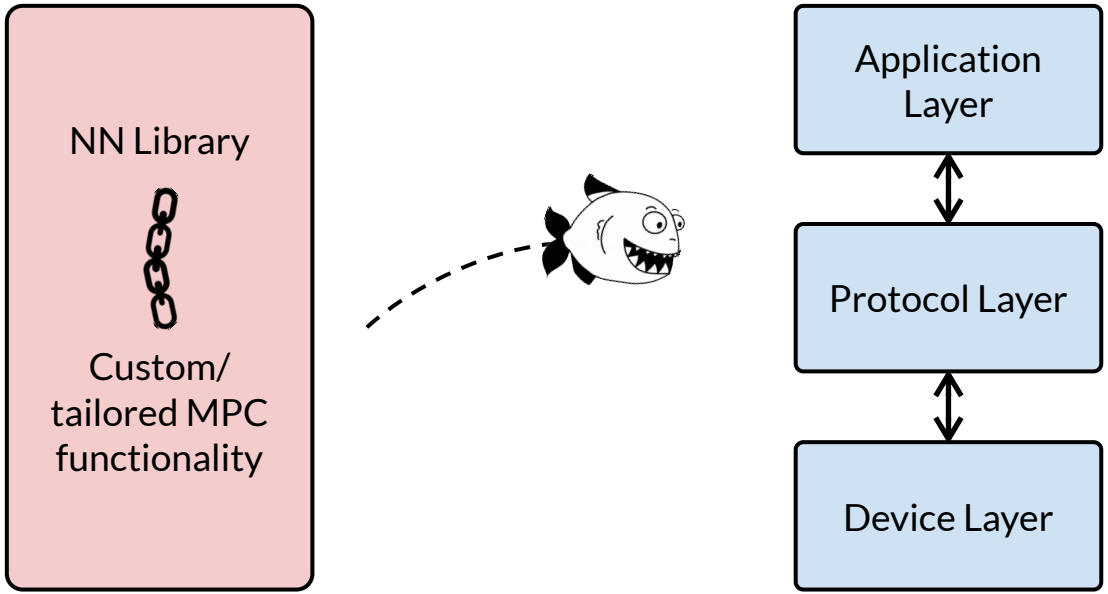
Creating a usable *platform* for MPC



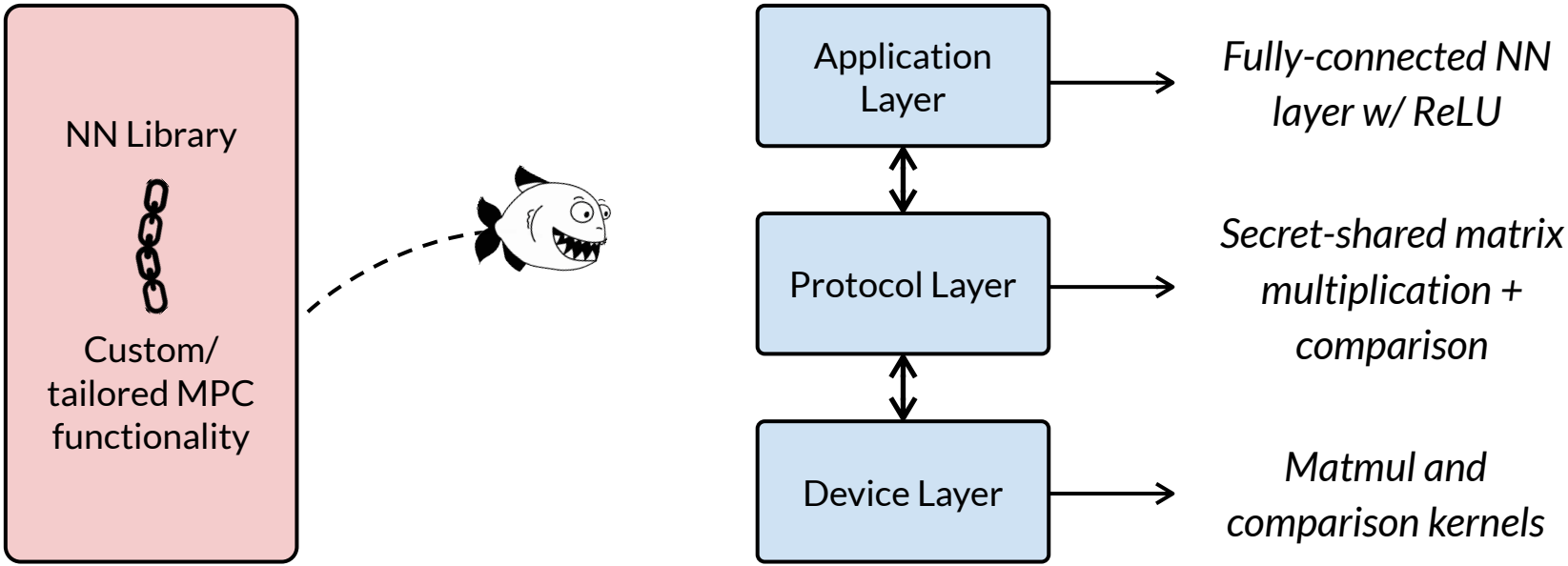
Monolithic

Creating a usable *platform* for MPC

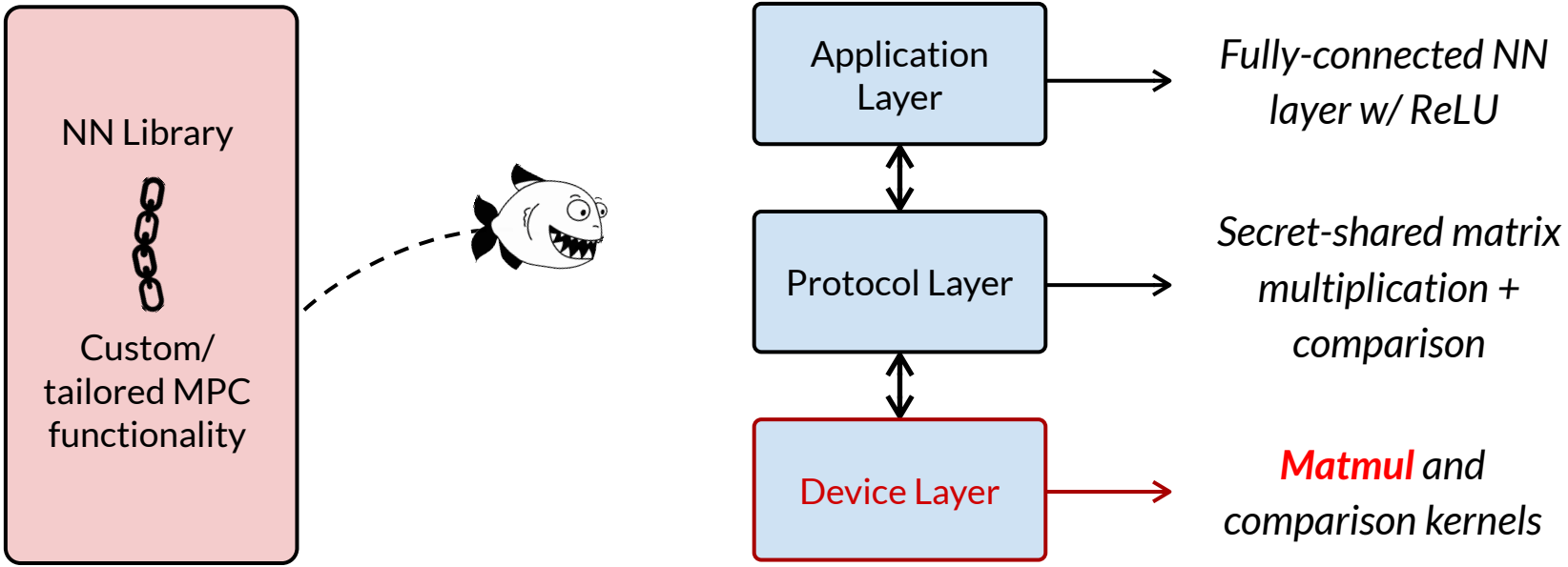
Piranha uses a modular approach to avoid redundancy and easily reuse MPC protocols in different settings.



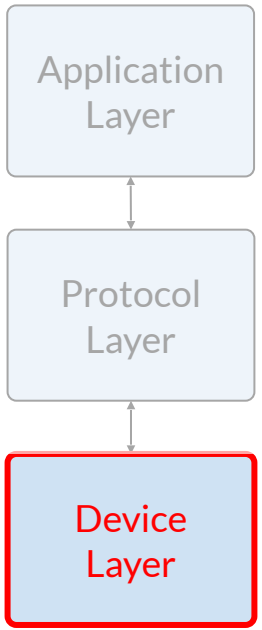
Piranha's architecture in practice



Problem 1: Performant linear operations for MPC



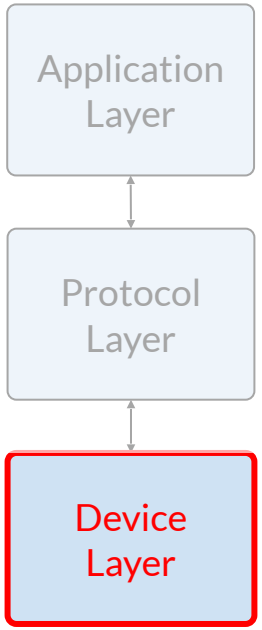
(1) Integer-based GPU acceleration is missing



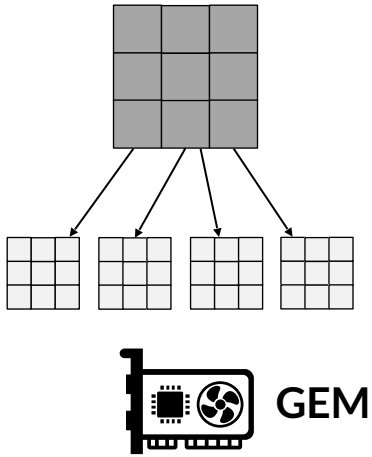
LSS protocols operate over integer rings and use *fixed point encoding* for ML training to encode real values.

Big issue: no performant kernels are available for integer GEMM (general matrix multiplication)

(1) Prior work adapts floating point kernels

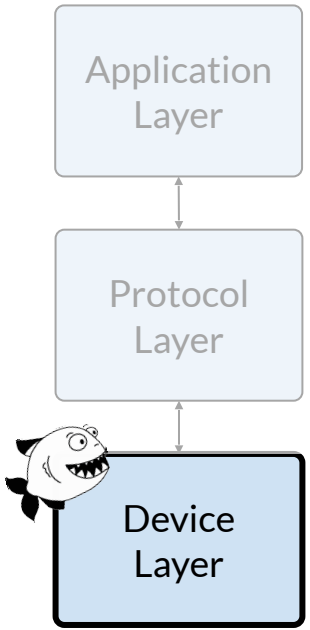


Prior work [TKT+21] splits 64-bit integers into 16-bit float chunks, incurring compute overhead.



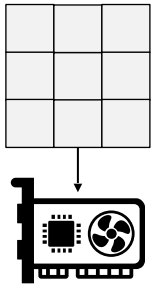
Assumes floating point performance outweighs overhead.

(1) Piranha directly uses GPU integer cores



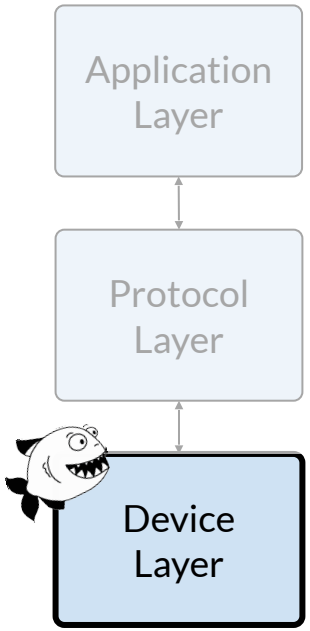
Piranha provides integer kernels directly to MPC protocols

We implement **32/64-bit integer** kernels with CUTLASS[1].



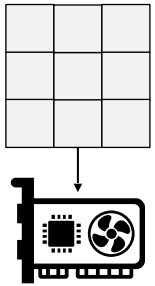
[1] https://github.com/NVIDIA/cutlass

(1) Piranha directly uses GPU integer cores



Piranha provides integer kernels directly to MPC protocols

We implement **32/64-bit integer** kernels with CUTLASS[1].

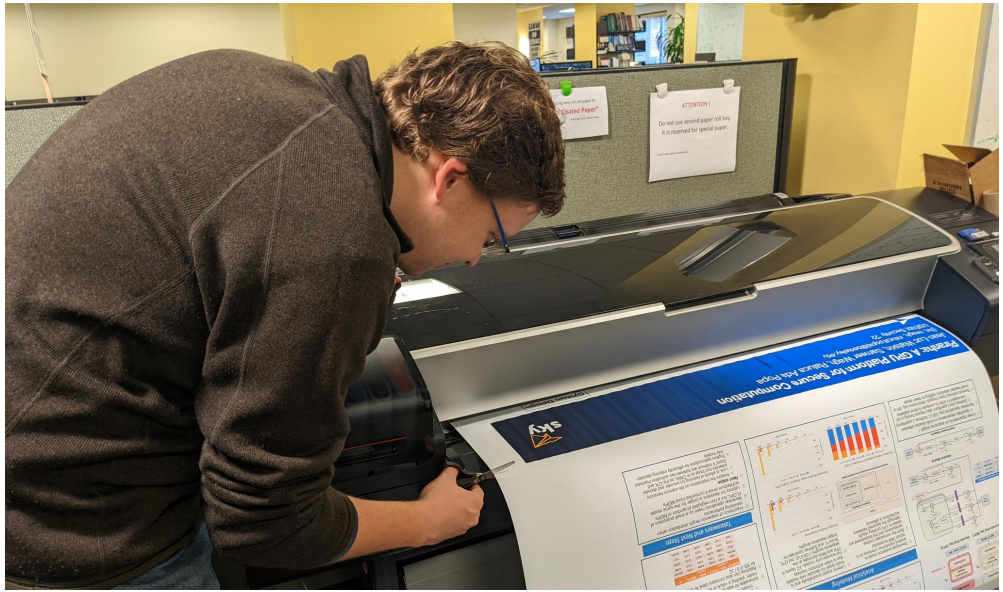


10x cuBLAS `f64`: 47 ms | Piranha `int64`: **4.9 ms**

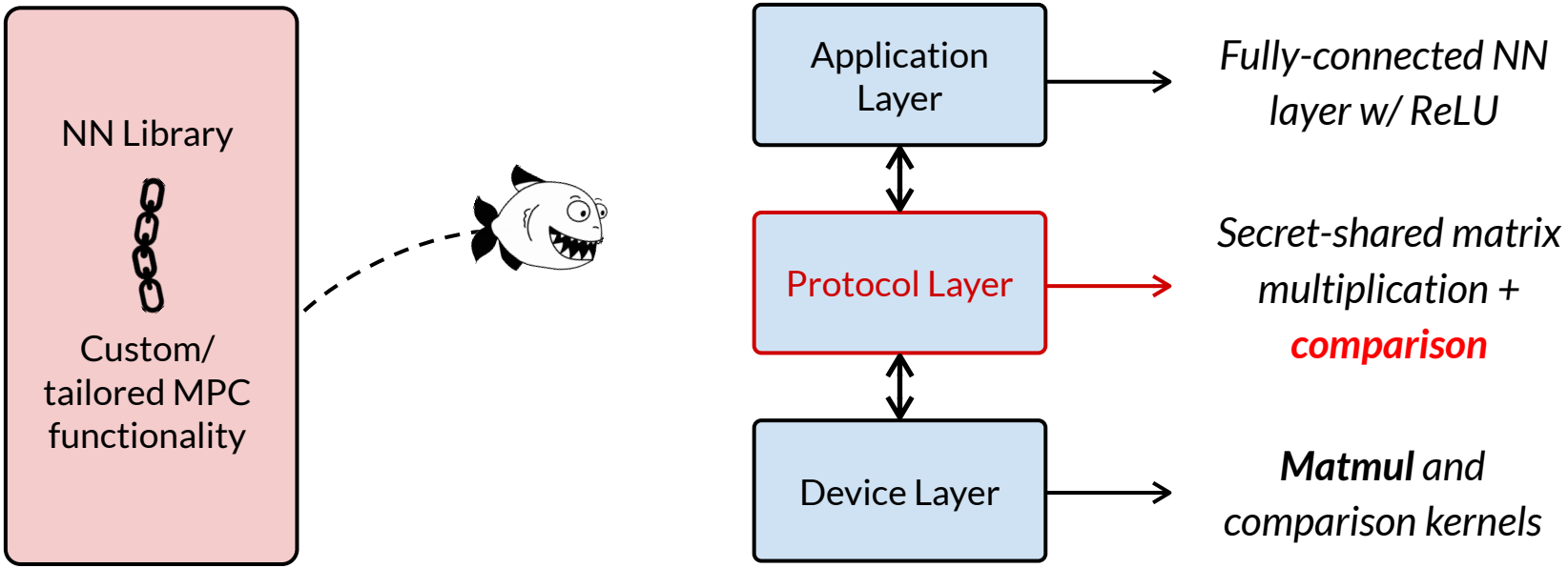
[1] https://github.com/NVIDIA/cutlass

(1) Piranha directly uses GPU integer cores

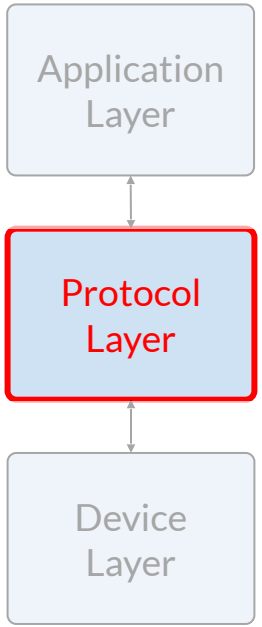
Lesson: make sure you're using the right tools for the job



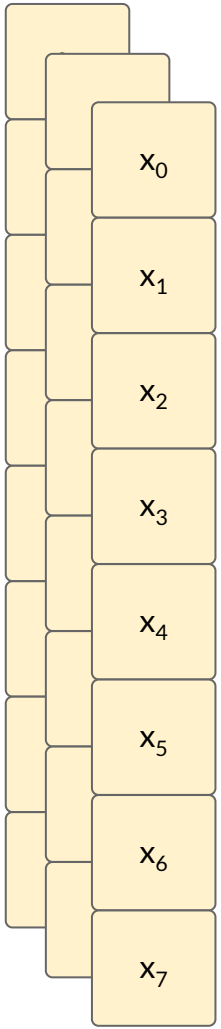
Problem 2: Memory-efficient comparisons



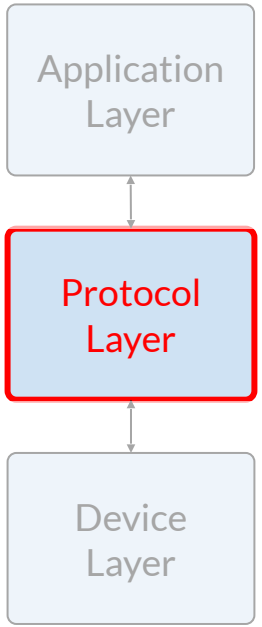
(2) MPC rapidly consumes GPU memory



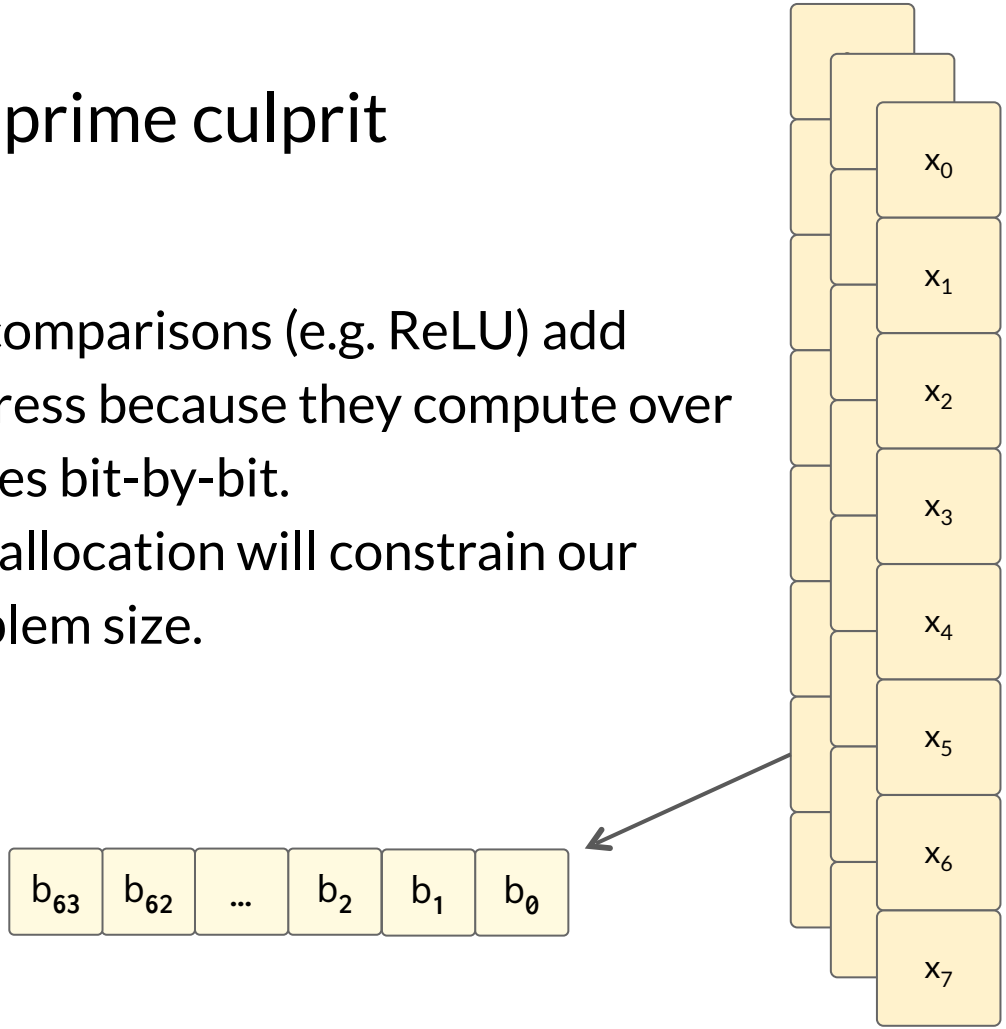
- **The issue:** Secret-sharing induces data duplication that stresses on-GPU memory.



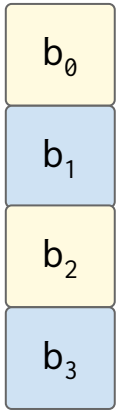
(2) Comparisons are the prime culprit



- Oblivious comparisons (e.g. ReLU) add memory stress because they compute over secret values bit-by-bit.
- Additional allocation will constrain our useful problem size.

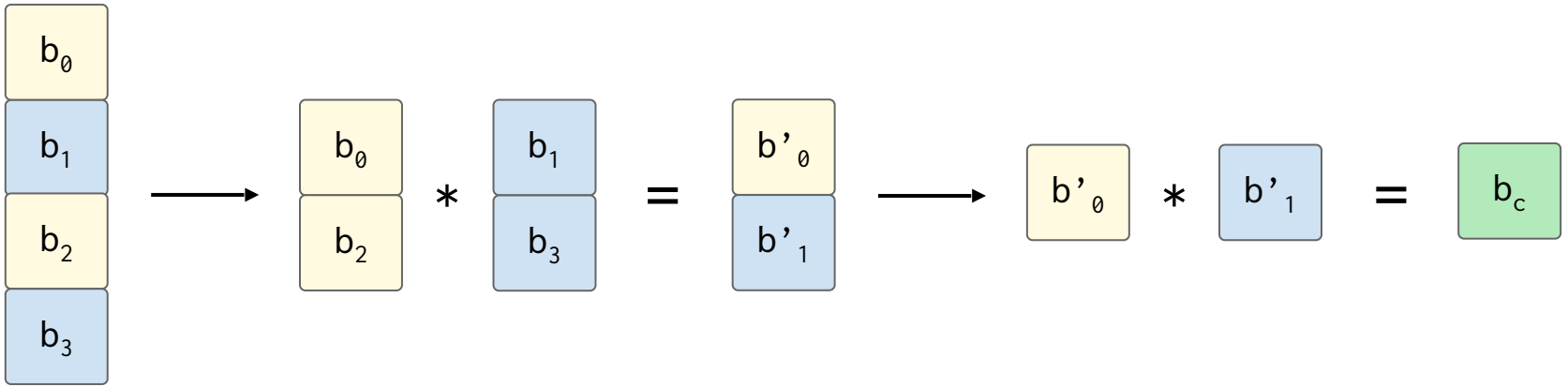


(2) Naïve string multiplication



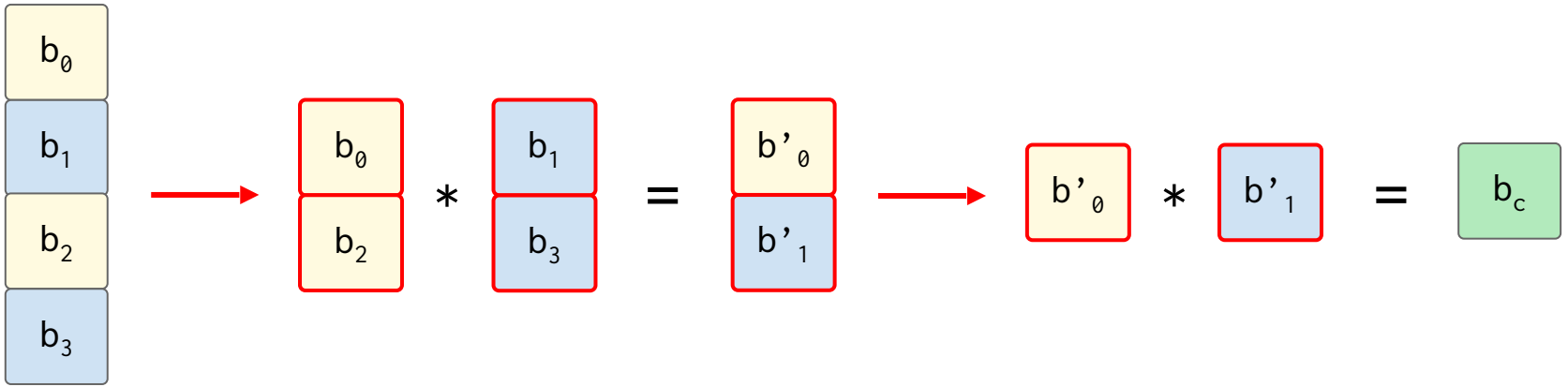
$$b_c = \prod_i b_i$$

(2) Naïve string multiplication



$$b_c = \prod_i b_i$$

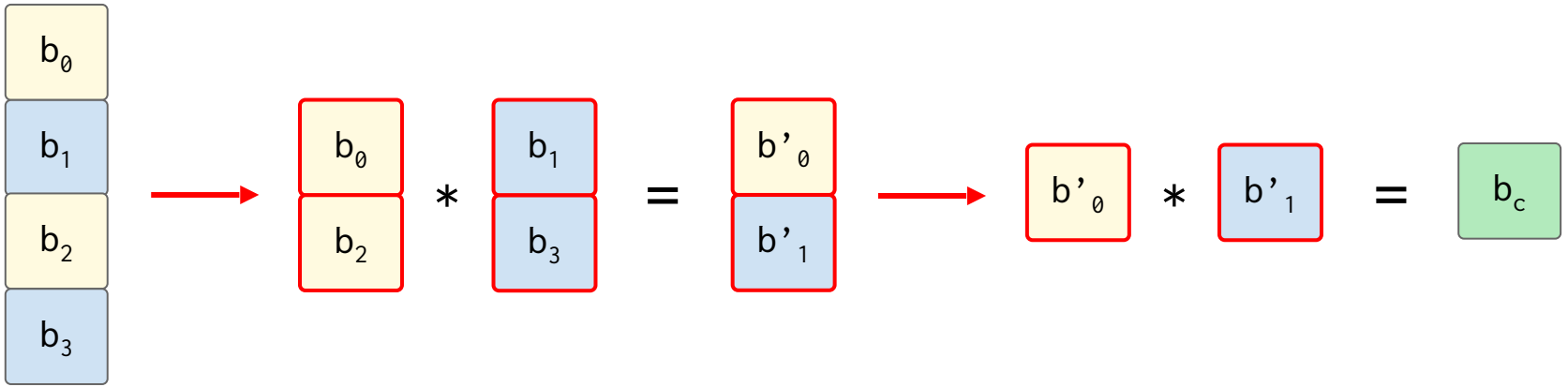
(2) The naïve protocol wastes memory



$$b_c = \prod_i b_i$$

(2) Iterator-based views keep memory in one place

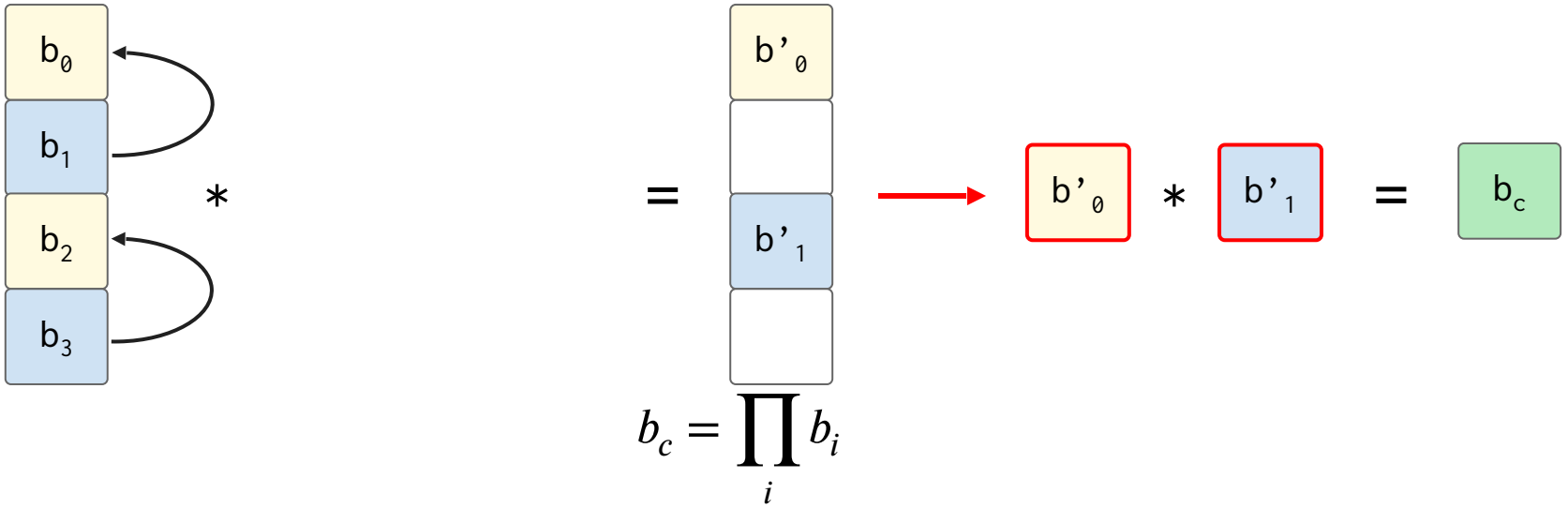
- Piranha allows protocols to use **iterator-based views for intricate data access patterns**:



$$b_c = \prod_i b_i$$

(2) Iterator-based views keep memory in one place

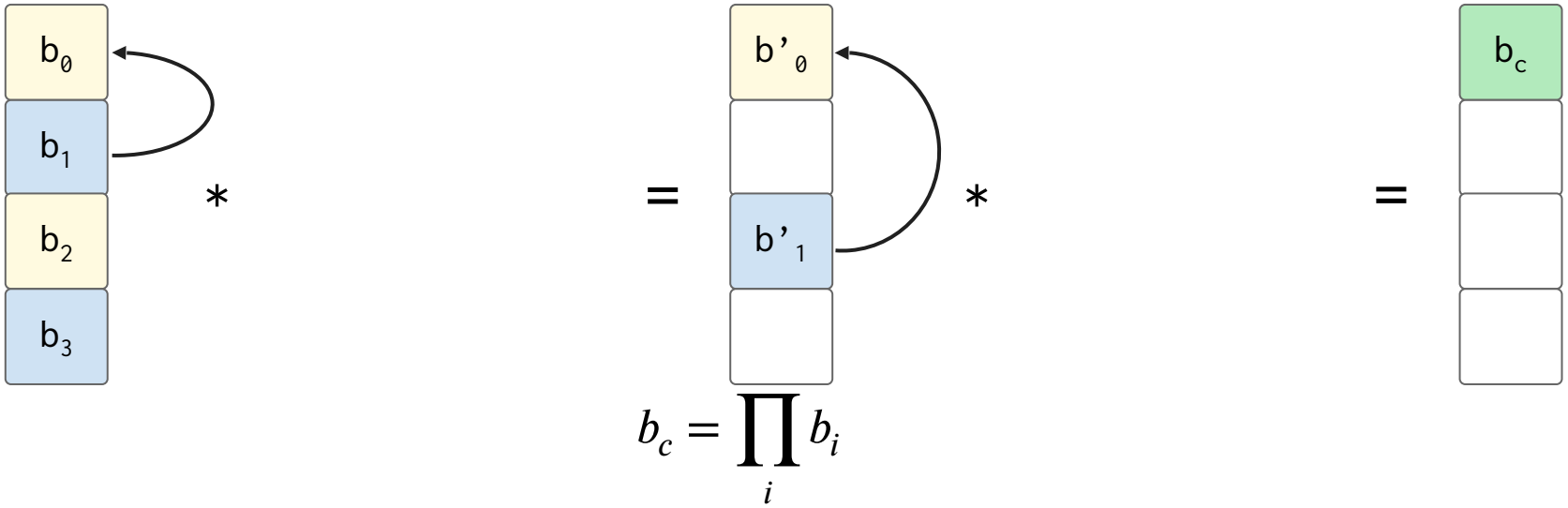
- Piranha allows protocols to use **iterator-based views for intricate data access patterns**:



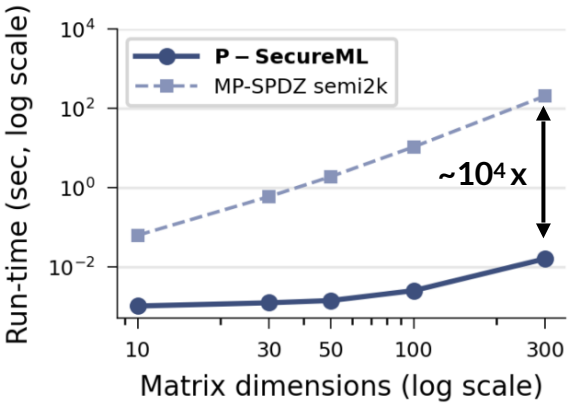
$$b_c = \prod_i b_i$$

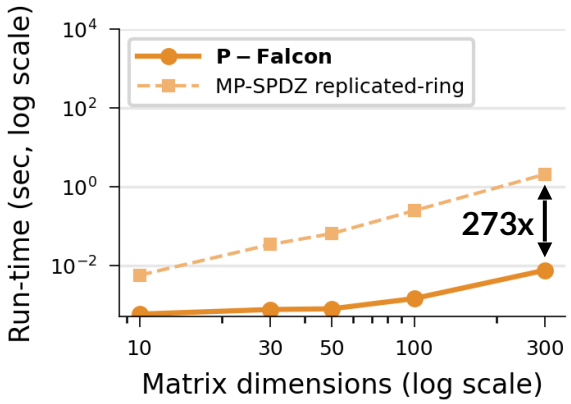
(2) Iterator-based views keep memory in one place

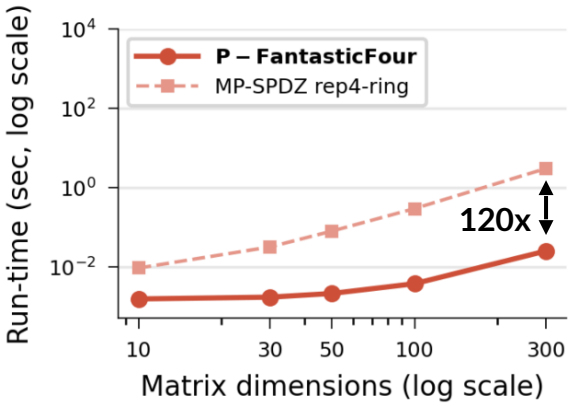
- Piranha allows protocols to use **iterator-based views for intricate data access patterns**:



Microbenchmarks: is Piranha *performant*?

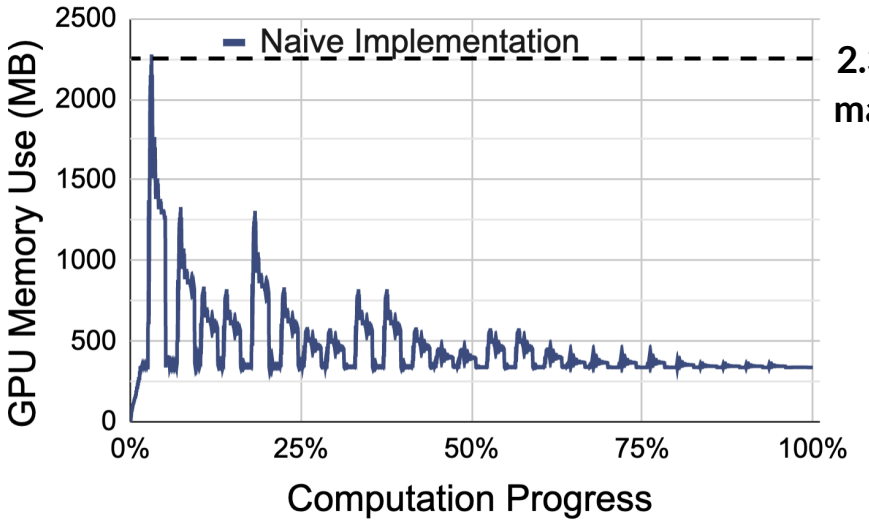






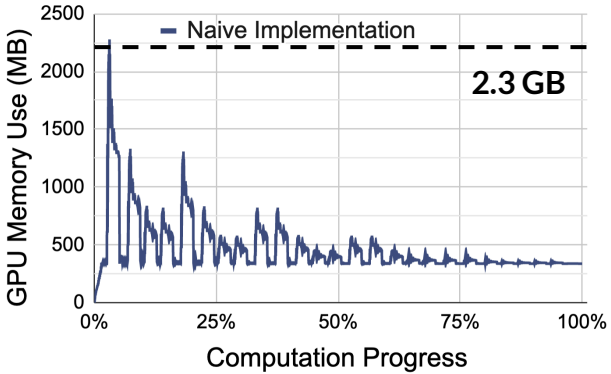
Piranha boosts performance by several orders of magnitude *across a range implemented MPC protocols.*

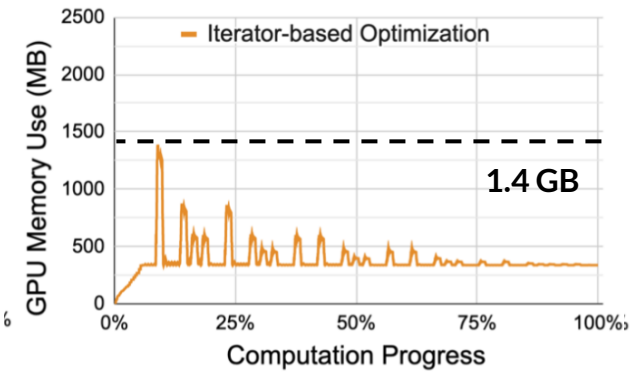
Memory Efficiency

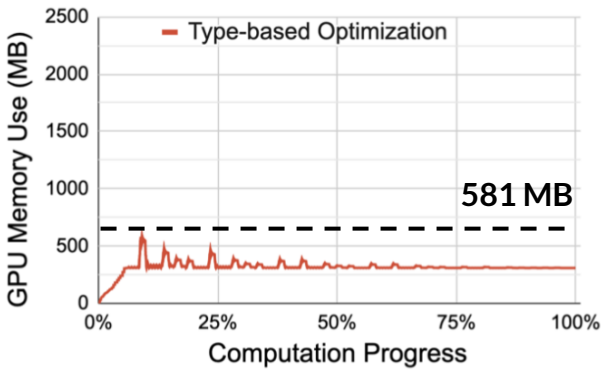


2.3 GB
max memory load

Memory Efficiency







Iterator-based and correct typing allows Piranha to *drastically reduce on-device memory consumption.*

End-to-end training: is Piranha *usable*?

Falcon estimated that the same training run would take it **14 days** on a CPU

Piranha accelerates a 3-party protocol to complete 10 epochs of VGG16 training in just **33** hours!

Network (Dataset)	Protocol	Time (min)	Comm. (GB)	Accuracy	
				Train (%)	Test (%)
SecureML (MNIST)	P-SecureML	12.99	49.55	97.37	96.56
	P-Falcon	7.51	22.84	97.37	96.56
	P-FantasticFour	23.39	33.01	97.37	96.56
LeNet (MNIST)	P-SecureML	87.55	683.18	96.78	96.80
	P-Falcon	71.56	485.90	96.88	97.10
	P-FantasticFour	219.20	676.13	96.88	97.11
AlexNet (CIFAR10)	P-SecureML	156.01	740.50	40.74	40.47
	P-Falcon	110.66	382.18	40.59	40.71
VGG16 (CIFAR10)	P-Falcon	1979.92	17235.35	55.13	54.26
	P-FantasticFour	7697.54	29106.24	55.02	54.35

Summary

Private ML training can feasibly leverage MPC without completely sacrificing performance by restructuring local share operations to move protocol execution to resource-constrained GPUs.

This talk

Metadata-hiding communication

[CGB+15, KCG+17, ECG+21, AS16, ...]

Mobile phones

Multi-party computation

[Yao86, GMW87]

Piranha

GPUs

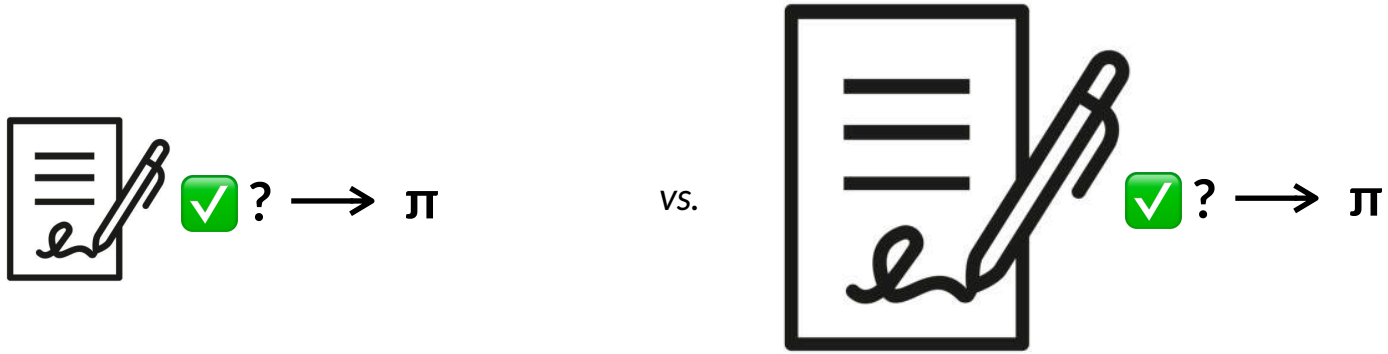
Zero-knowledge Proofs

[FFS87]

ZK Proofs at Scale

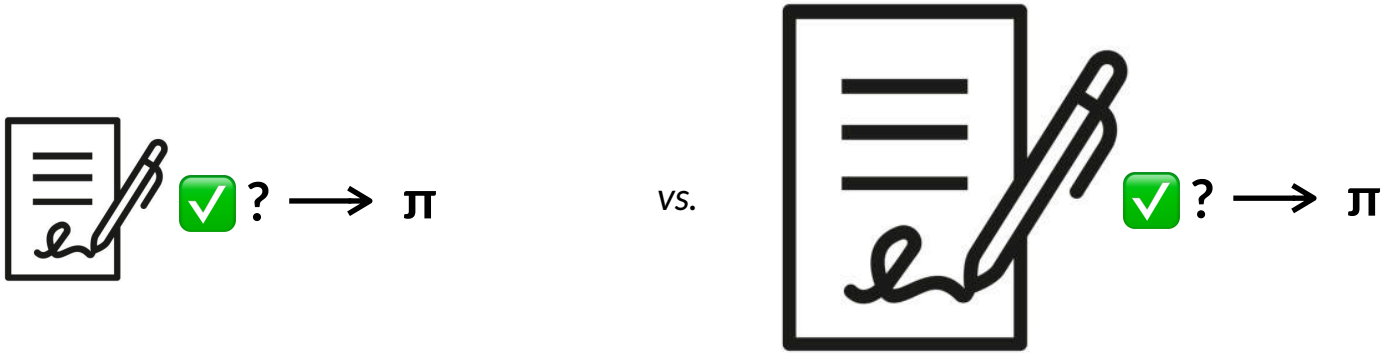
GPUs

Scaling zero-knowledge proofs



- Piranha designed a platform for on-device training and optimized memory usage within the confines of a single GPU

Scaling zero-knowledge proofs



- Piranha designed a platform for on-device training and optimized memory usage within the confines of a single GPU
- Zero-knowledge proofs are even more memory-intensive (many GBs per proof)

What happens when we run out of GPU memory entirely?

Focus on a critical bottleneck (~80%): multi-scalar multiplication

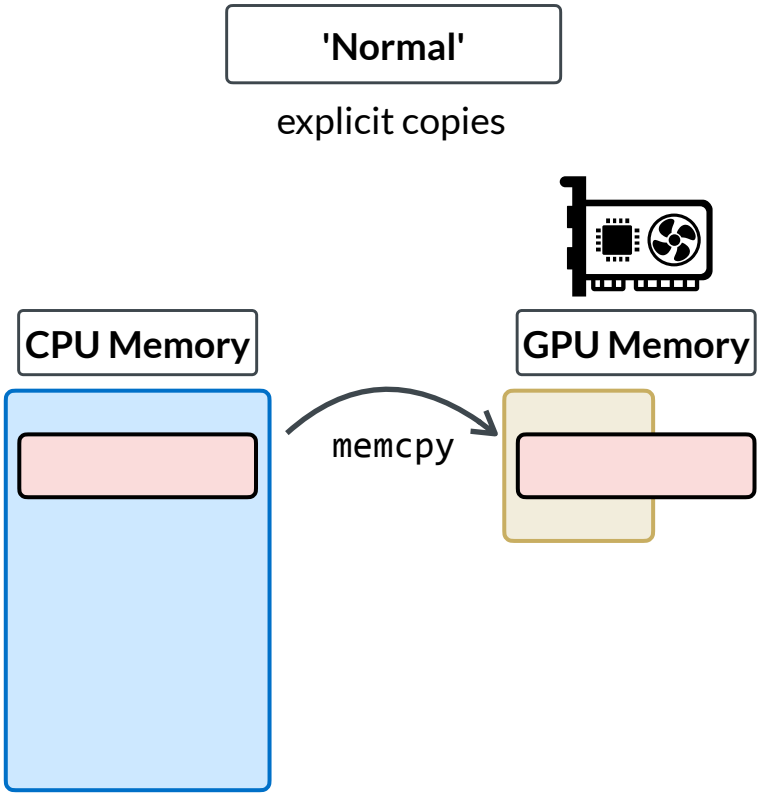
$$\sum_{i=1}^{n} k_i P_i$$

scalar

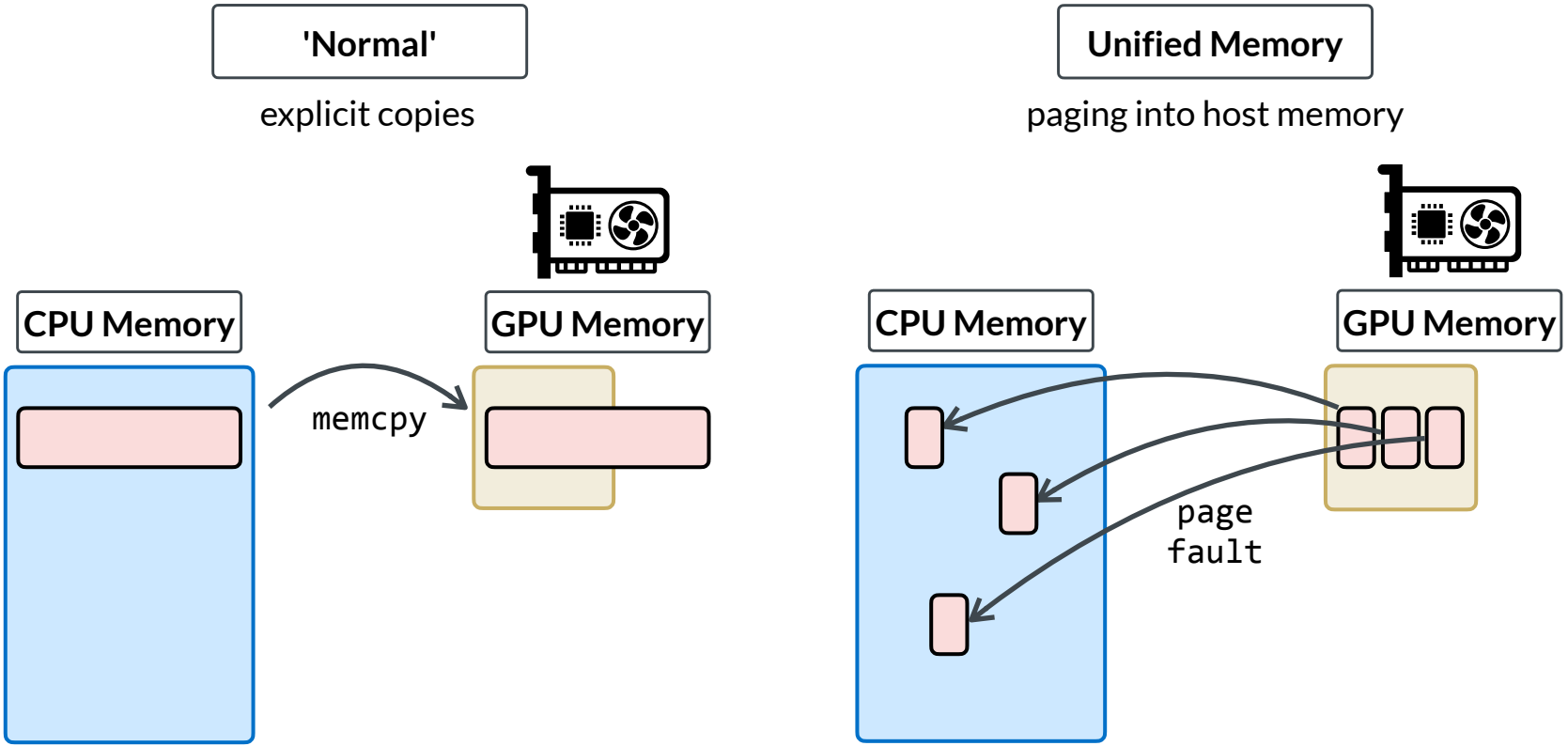
elliptic curve point

- MSM size scales as a function of the circuit parameters
  - (e.g. 2^{20} for a single signature verification)

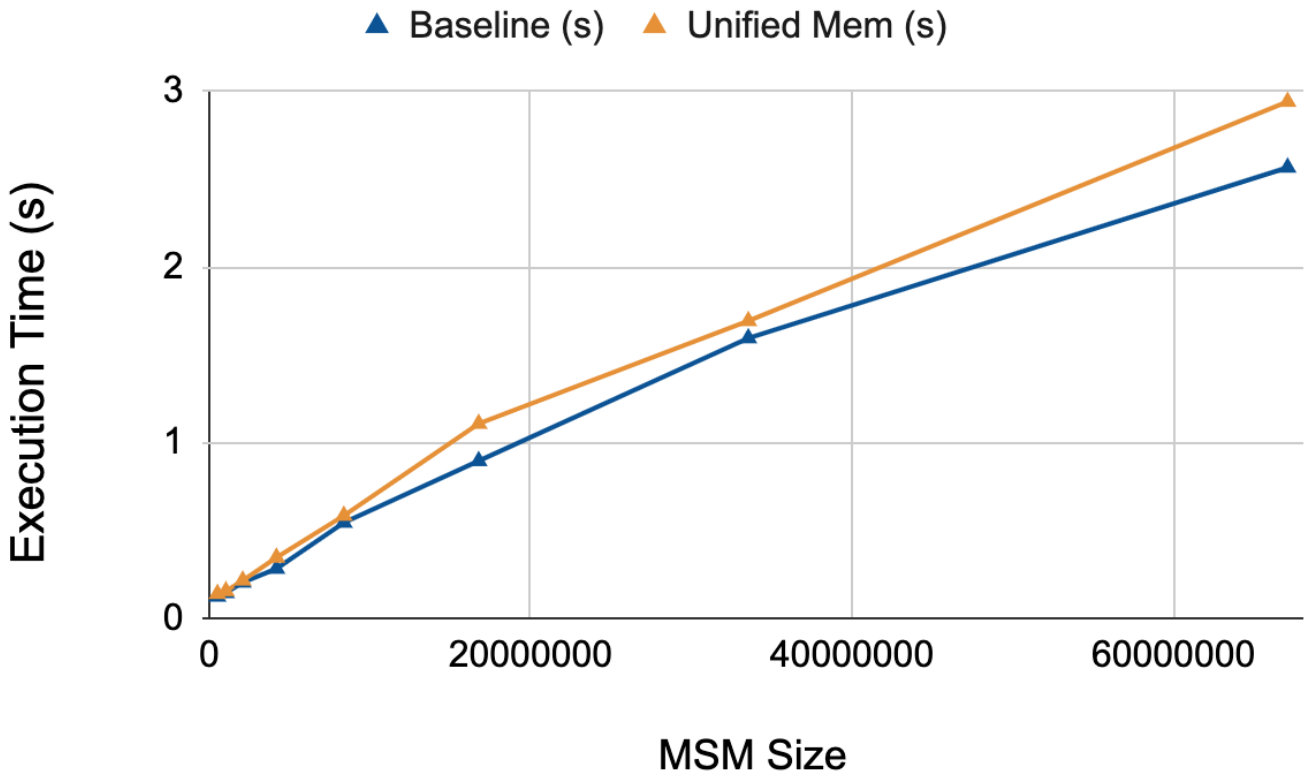
Simple idea: leverage unified memory to increase scale



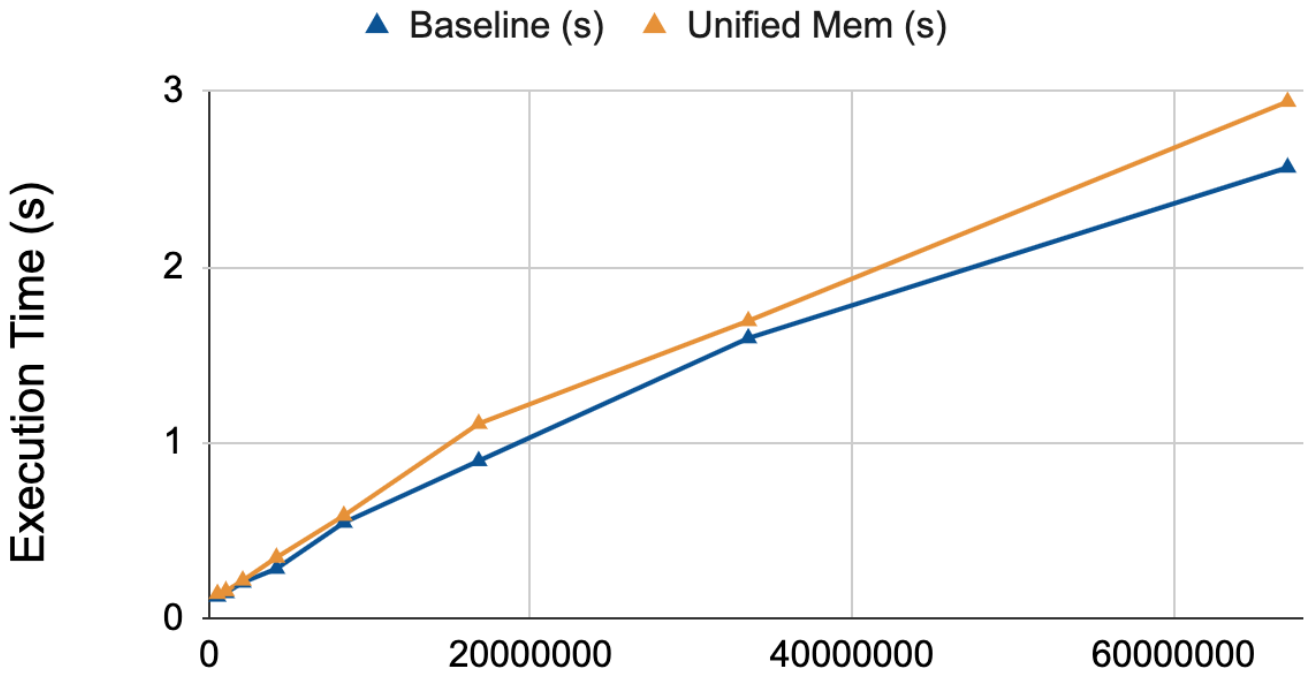
Simple idea: leverage unified memory to increase scale



Now we see a small but persistent paging overhead

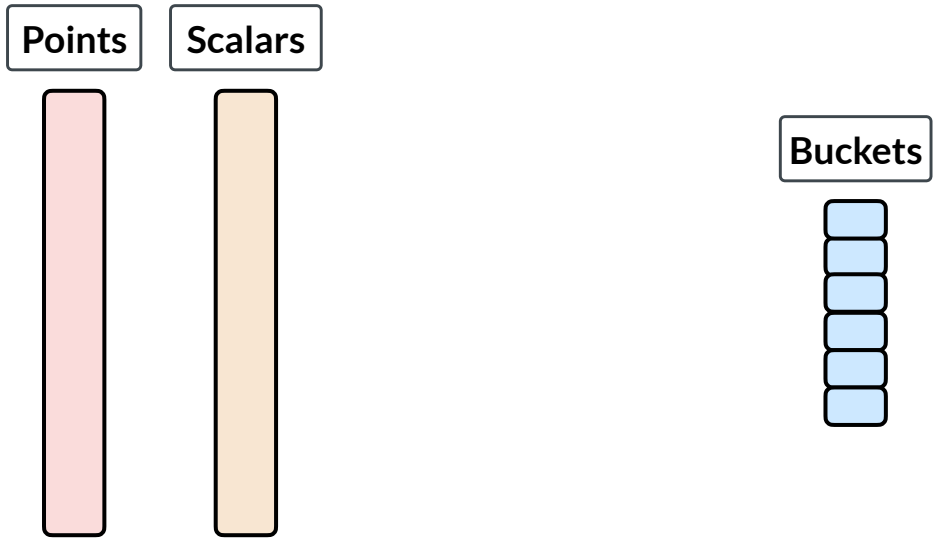


Now we see a small but persistent paging overhead

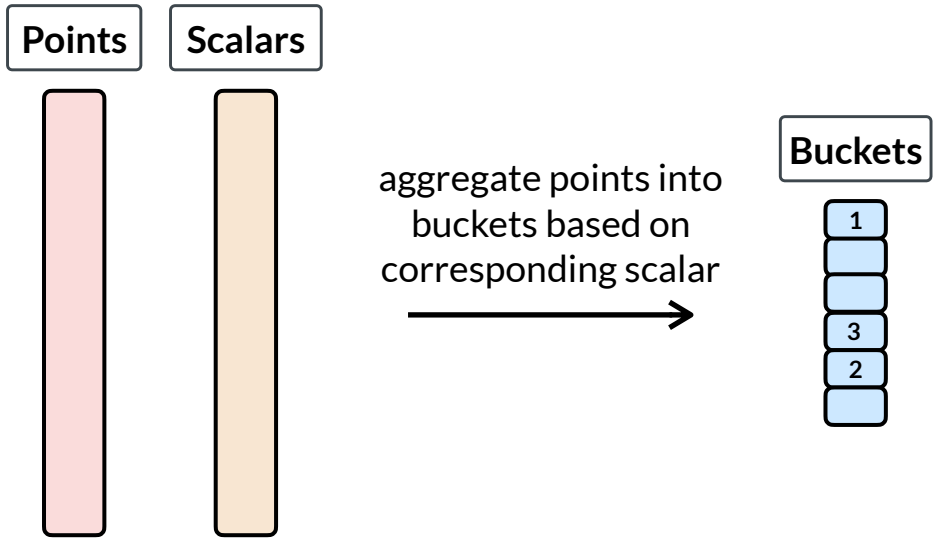


Can we do better?

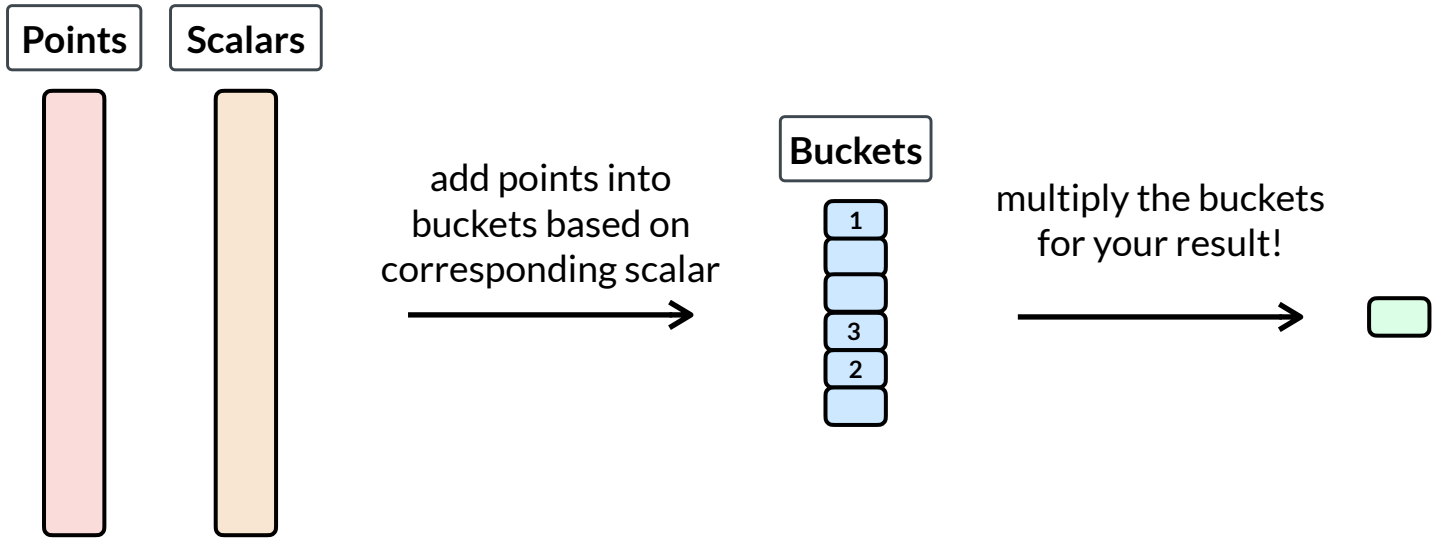
How does an MSM work?



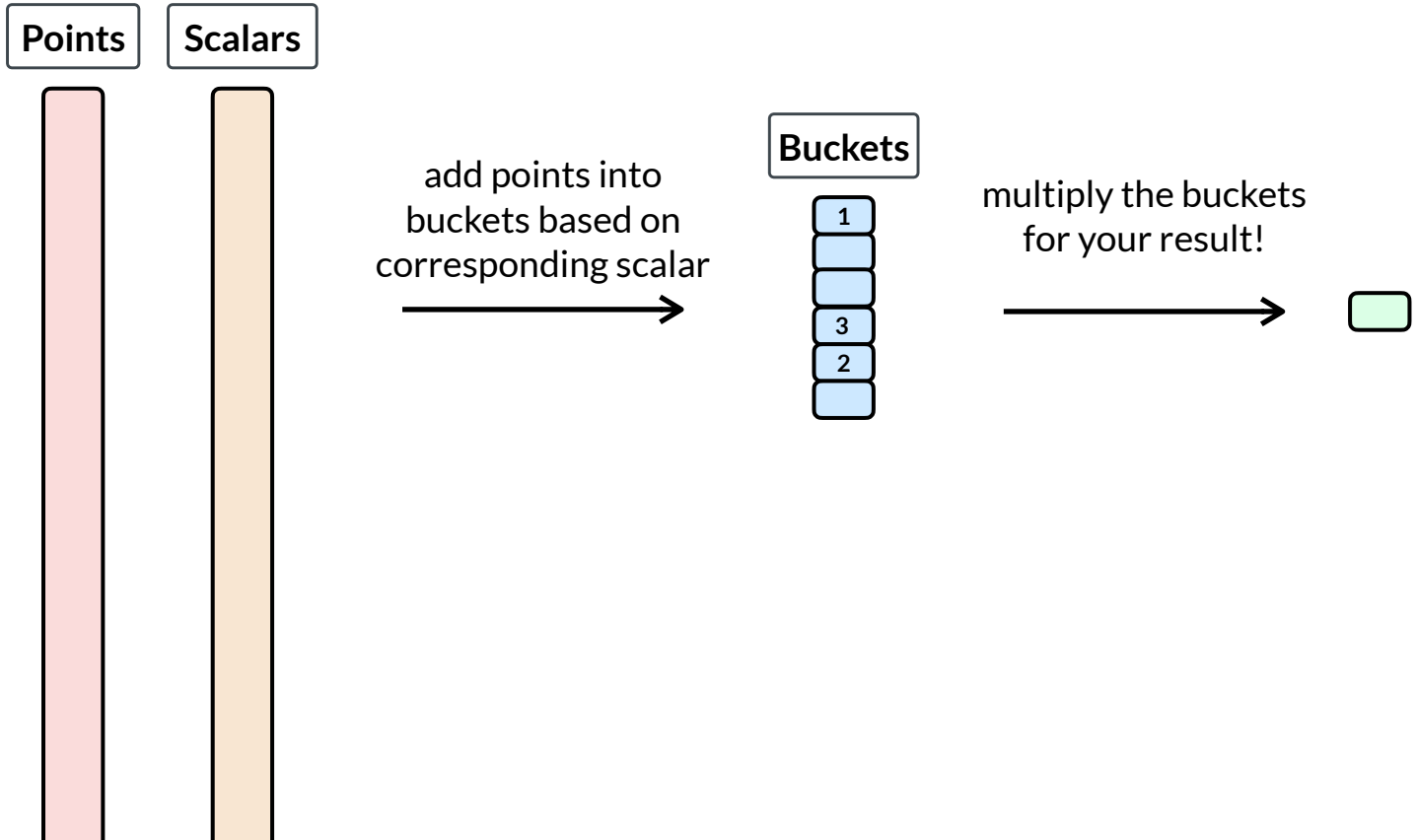
How does an MSM work?



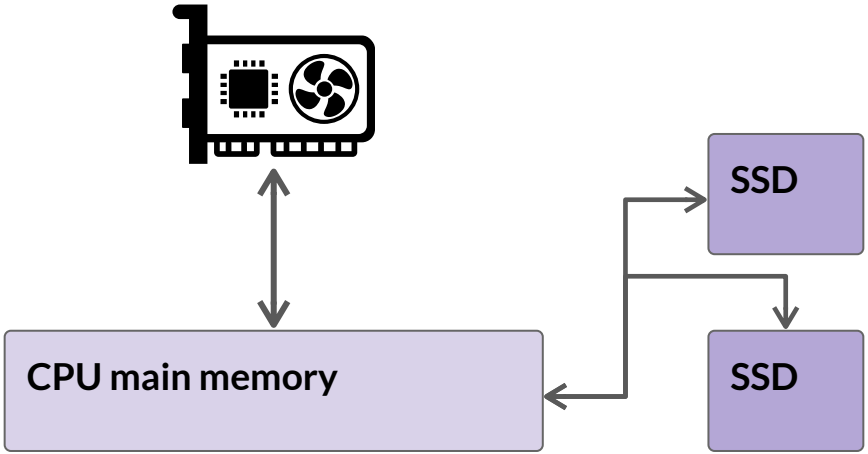
How does an MSM work?



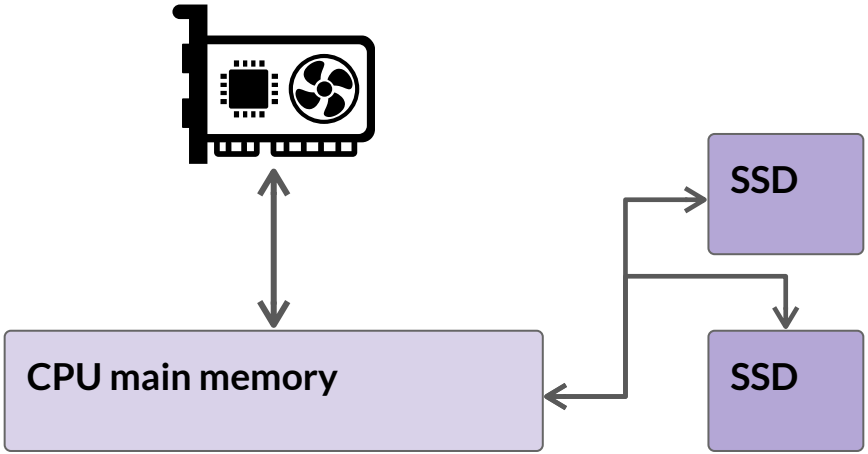
How does an MSM work?



Idea: spill GPU memory to both CPU and disk

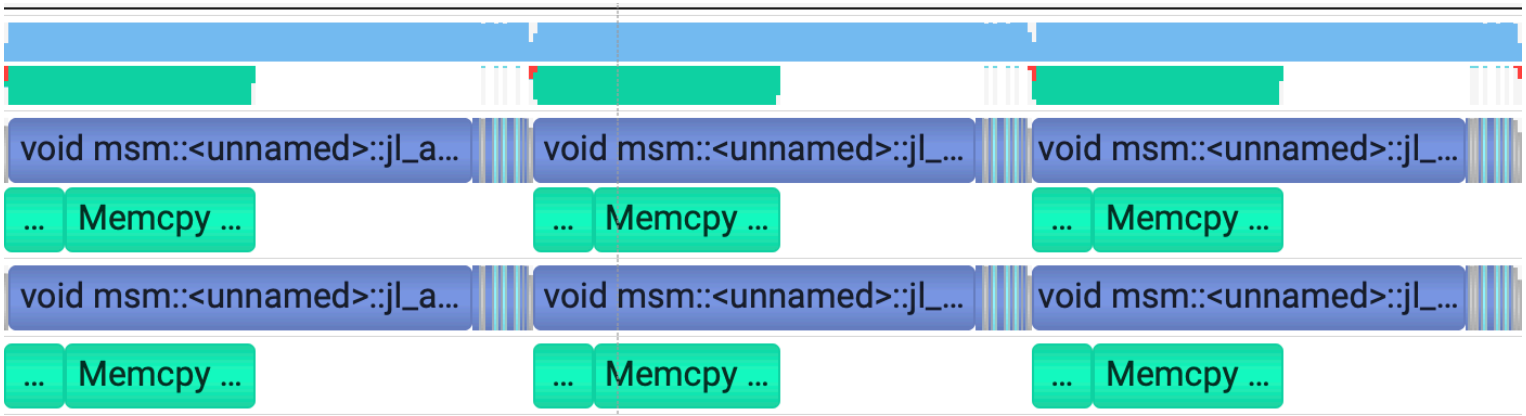


Idea: spill GPU memory to both CPU and disk

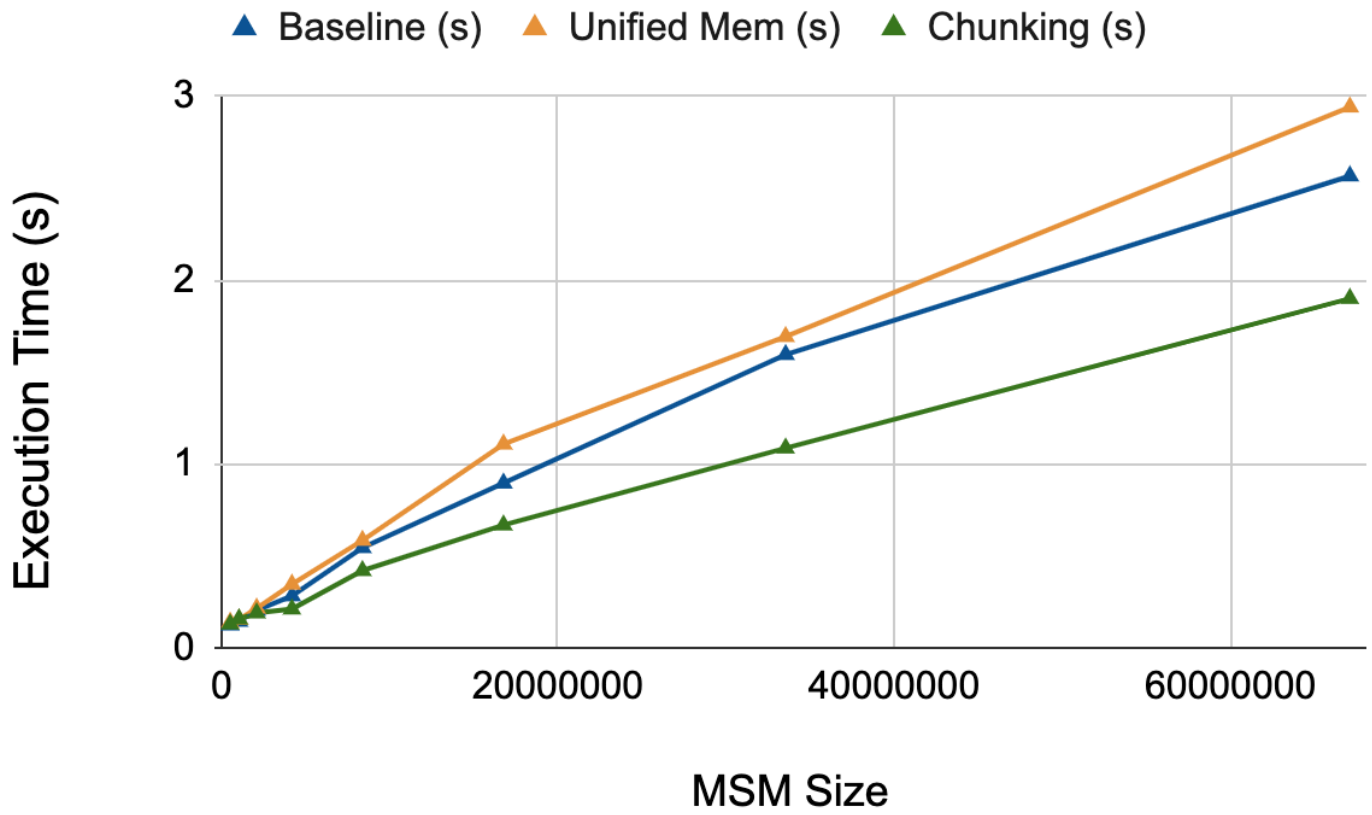


Hypothesis: slow computation time of chunked MSM segments can hide memory access latency for next chunk, allowing effectively unbounded-size problems

Bucket aggregation is slow and allows memory movement



Chunking is better for performance!



Summary

Complex attestation can feasibly leverage ZKPs without completely sacrificing performance by restructuring multi-scalar multiplication to move protocol execution to resource-constrained GPUs.

Co-designing Cryptographic Systems with Resource-Constrained Hardware

Jean-Luc Watson

Dissertation Talk
5/7/24