Veracruz

Privacy Preserving Multiparty Computation

29 April 2021

Who we are

Dominic Mulligan
[he/him]
Veracruz team member

dominic.mulligan@arm.com

arm

Derek Miller
[he/him]
Veracruz team member

derek.miller@arm.com

arm

Shale Xiong
[he/him]
Veracruz team member

shale.xiong@arm.com

arm

Christopher Haster
[he/him]
Veracruz team member

christopher.haster@arm.com

arm

CONFIDENTIAL COMPUTING CONSORTIUM

Background

A prevailing trend in the semiconductor industry is the addition of dedicated hardware support for **Confidential Compute**

Various names for these new hardware features: **Secure Enclaves**, **Realms**, **Protected Virtual Machines**

These protection mechanisms are also accompanied by a **Remote Attestation** procedure which allows a skeptic to challenge authenticity of an isolate

Question: if we assume isolates are widely deployed, what new distributed systems can we build with them?

The Veracruz framework

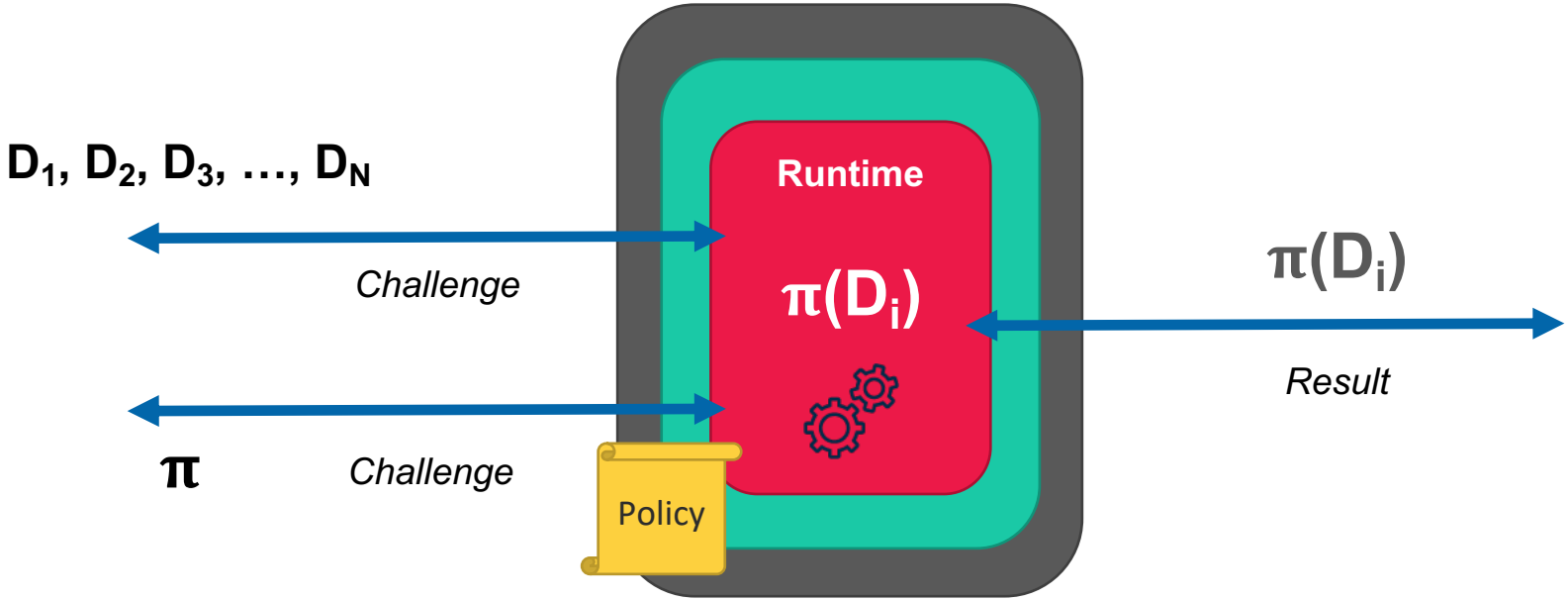
Veracruz is a **framework** for defining **privacy-preserving collaborative computations** using isolates as a neutral ground for computation

Computations defined using Veracruz consist of:

- *N* mutually distrusting **data providers** providing inputs to the computation,
- A single **program provider** providing the computation to use,

A **policy** describes identities and roles of principals, restricts access to result

Outline of Veracruz



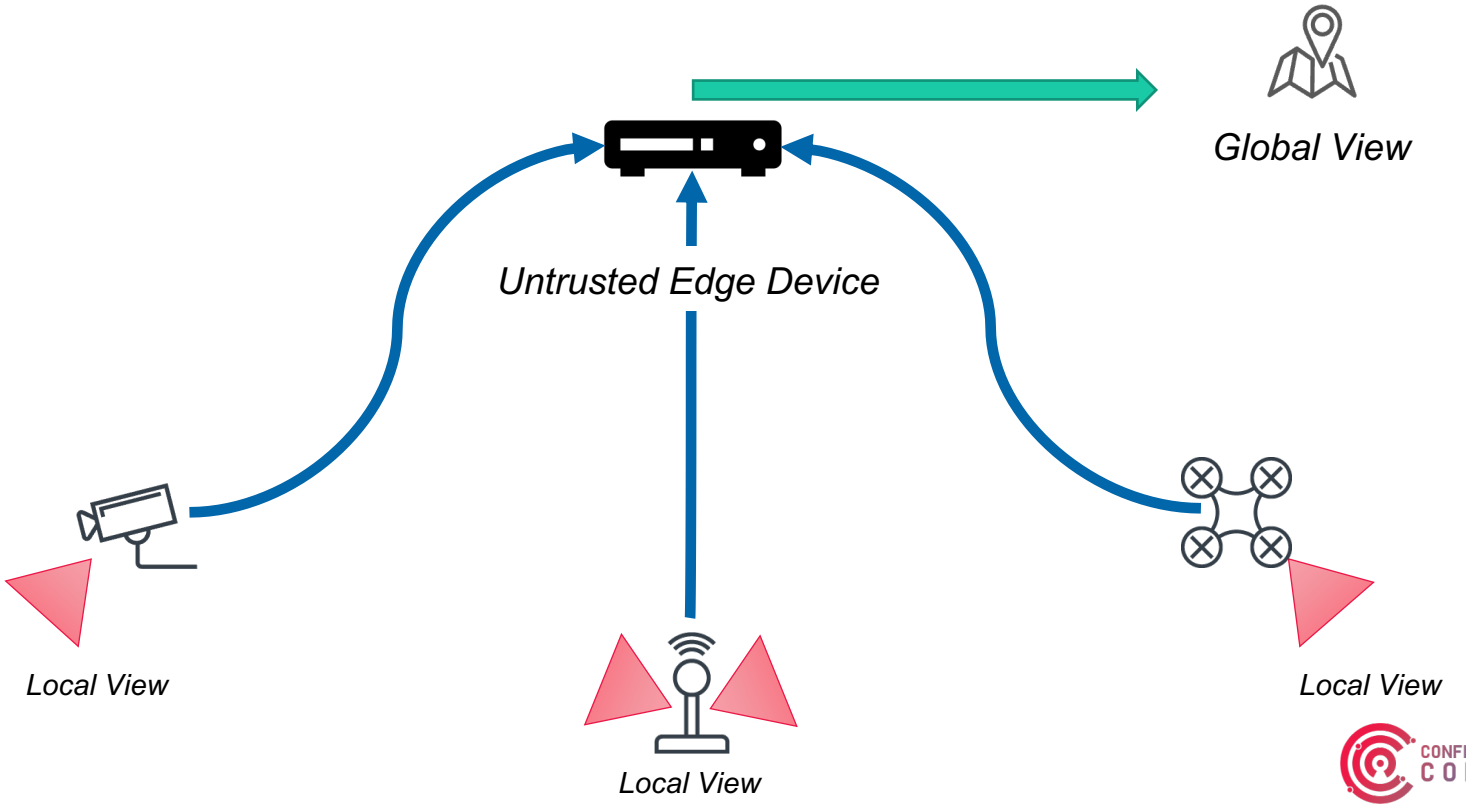
The Untrusted Host…

Spawns an Isolate…

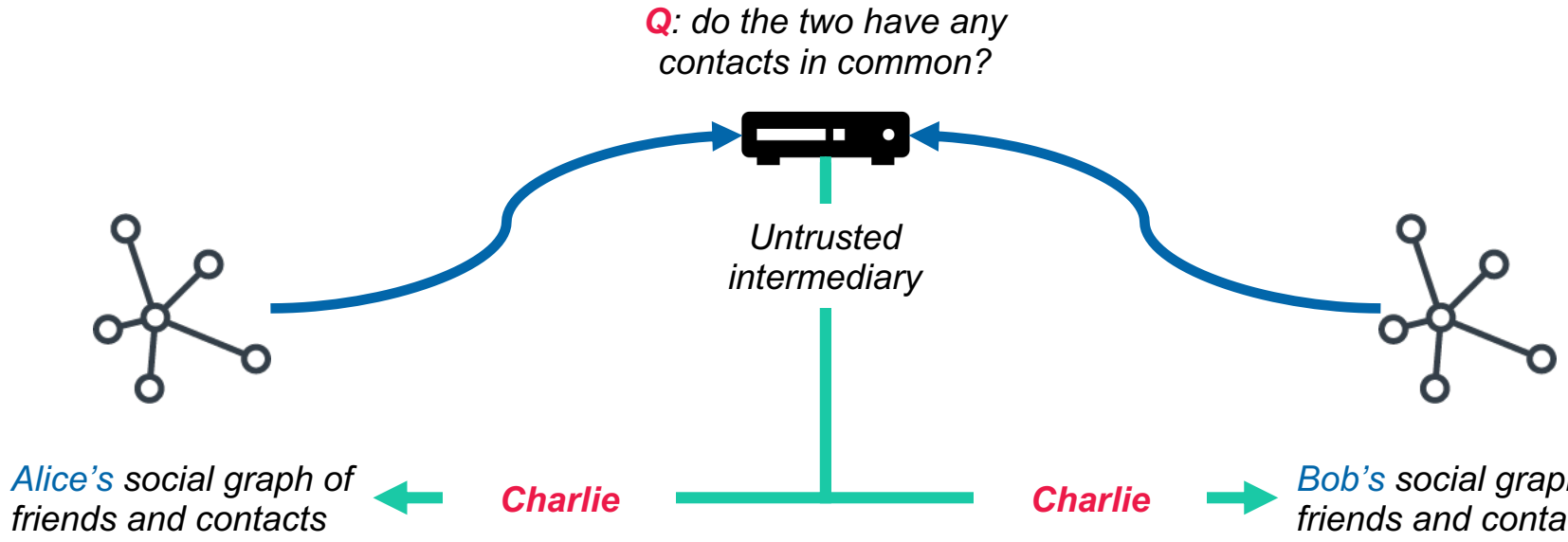
Hosting the Veracruz Runtime

CONFIDENTIAL COMPUTING CONSORTIUM

Example use-case: secure sensor aggregation



Example use-case: private social graph analytics



Many more use-cases…

- Private set intersection, and private intersection set-sum,
- Privacy-preserving machine learning,
- Protecting ML algorithm IP,
- Securely offloading large computations from computationally weak devices,
- Private auction, polls, and surveys,
- Oblivious routing in maps …and many more.

Observation: pattern captured by Veracruz appears over-and-over…

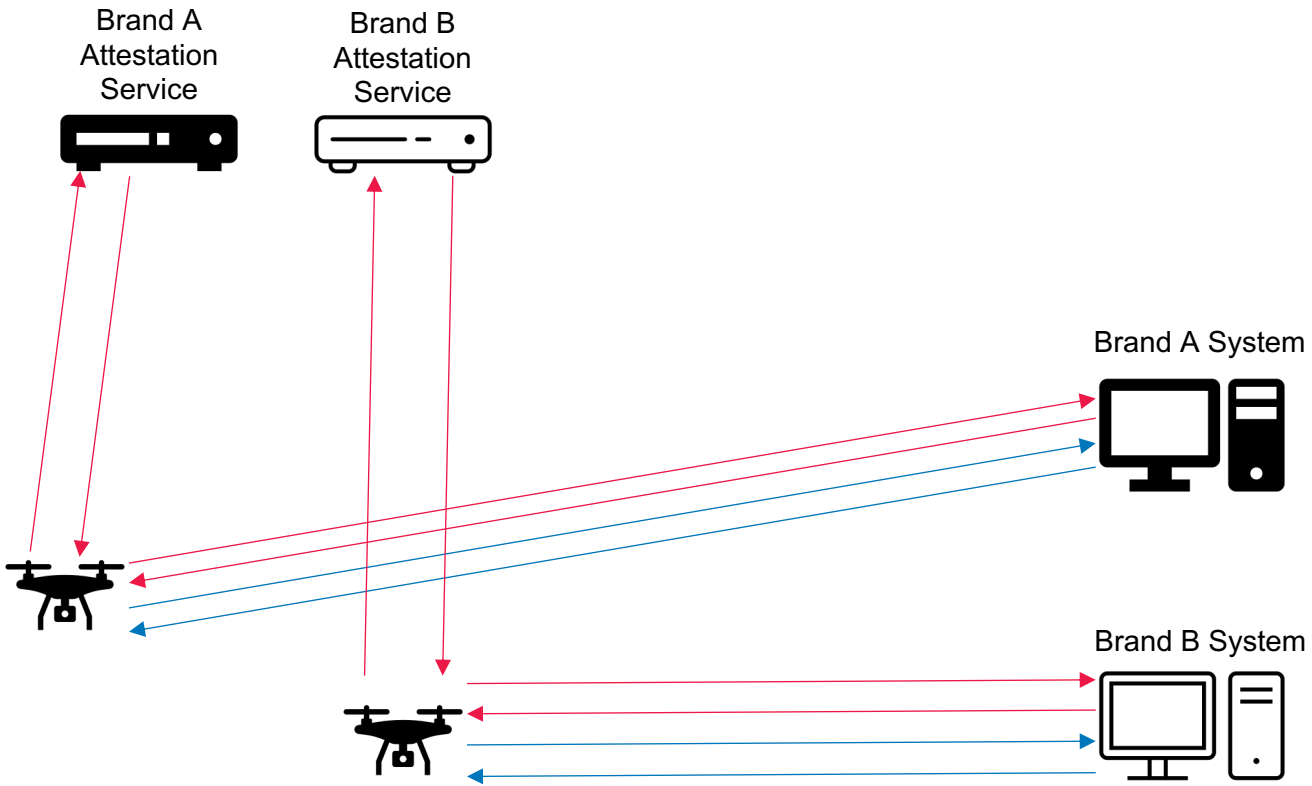
In the remainder of this talk...

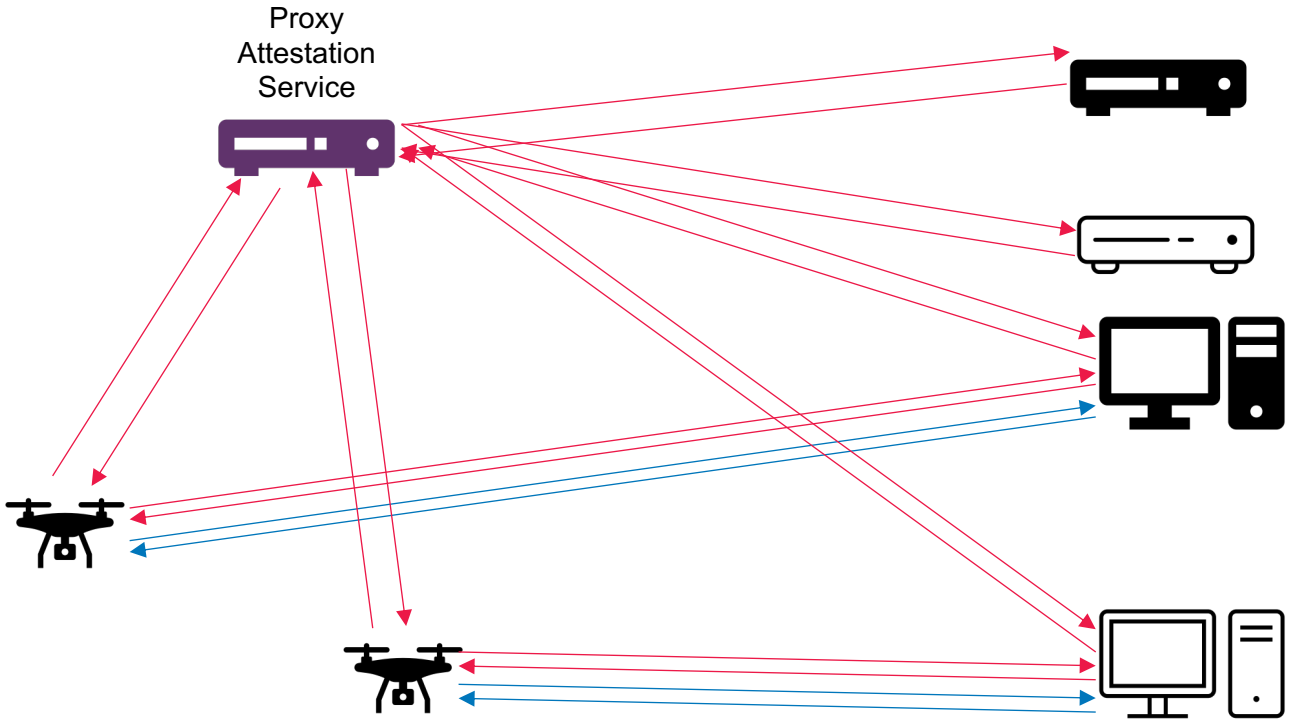
- Veracruz's use of attestation – Derek Miller
- Focus on the Veracruz Runtime – Shale Xiong
- Embedded clients, microcontroller-scale – Chris Haster
- Conclusions, further work, questions – Dominic Mulligan

Attestation

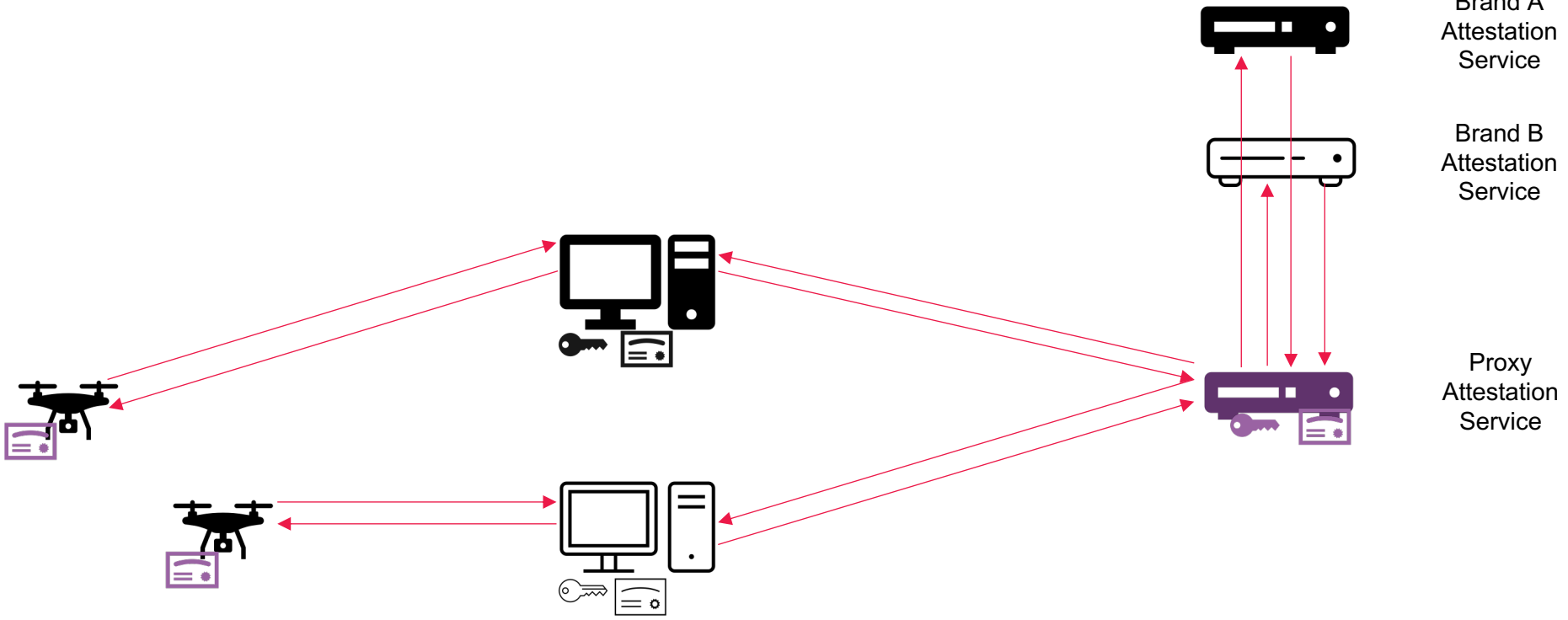
What is attestation?

- A way to establish trust in the state of a computing environment
  - The software being run
  - The software used during boot
  - the configuration
- Lots of different protocols (some more complicated than others)
  - Intel EPID
  - TPM
  - PSA Attestation
  - AWS Nitro Attestation
  - Often relies on an external third party to authenticate signed tokens (aka documents)

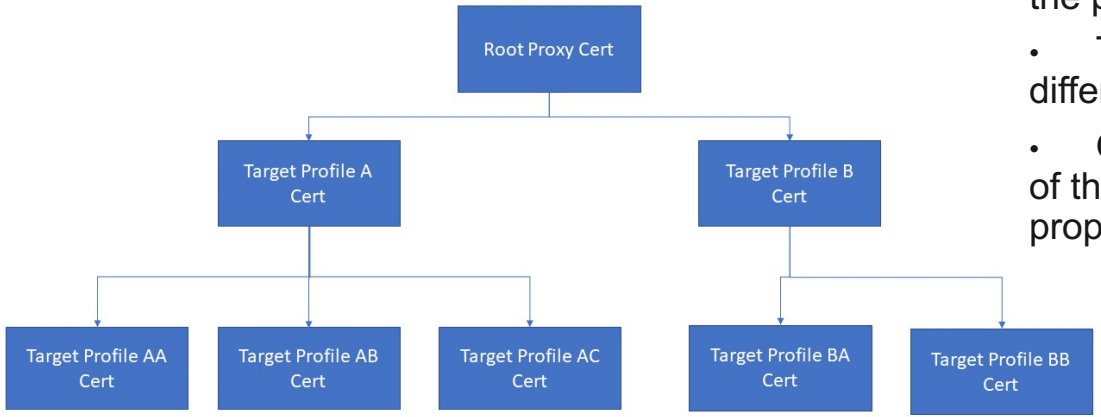

Brand A
Attestation
Service
Brand B
Attestation
Service
Brand A System
Brand B System



CONFIDENTIAL COMPUTING
CONSORTIUM



How to allow the client to select a platform profile?

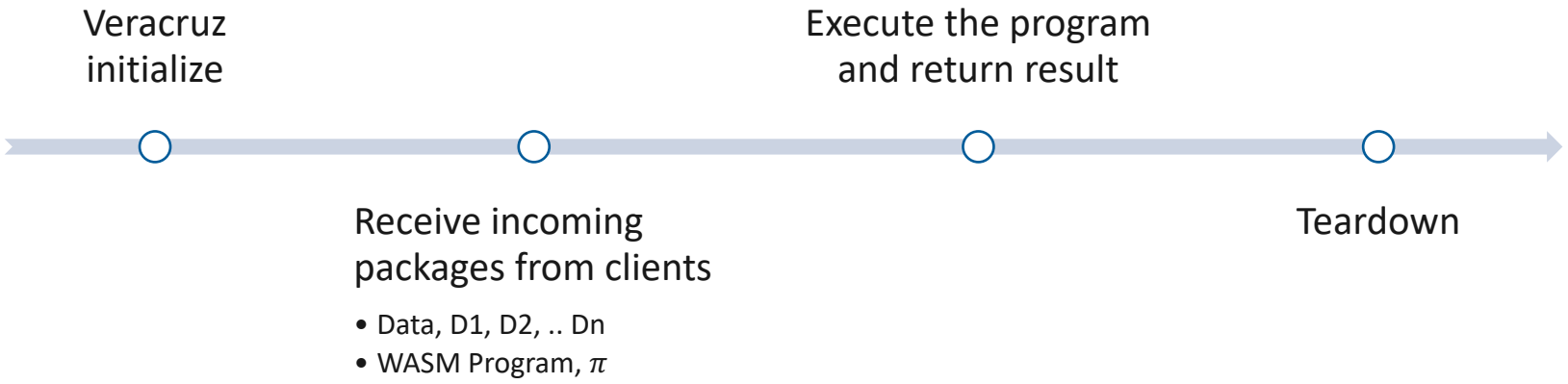


- The Proxy service can select which certificate/key pair to use based on the profile of the platform being attested
- This can be a list of certificates, each for a different profile
- Or it can be a tree of certificates, each level of the tree representing a different platform property

Veracruz Runtime (WASI and VFS)

Existing single-use runtime

The WASM program is loaded and executed by the Veracruz runtime **only once**.



Existing buffer-based storage and API

Disadvantage #1: More code. Veracruz runtime has separate and complex processes for incoming program and data packages from clients.

Disadvantage #2: Unfamiliar API. Veracruz runtime stores input and output in several buffers and provides API for the WASM program, for example:

- `__veracruz_hcall_read_input(index, size)`
- `__veracruz_hcall_write_output(buf, size)`

Limitations

- **Usability**. Program providers must work with unfamiliar API.

- **Setup cost**. Veracruz runtime setup time can be high. Ideally, Veracruz clients can reuse the same runtime for different tasks.

- **Streaming**. Many use cases, for example, secure sensor aggregation, requires to call the program multiple times on data streaming into the runtime.

Virtual filesystem

New proposal: A virtual filesystem (VFS) within the Veracruz instance (inside TEE) manages incoming data and programs, and results. The VFS implements some the WebAssembly System Interface (WASI).

We implement **file system related API in WASI**, for example,

- `path_open(…)`,
- `fd_read(…)`,
- `fd_write(…)`, …,

while ignore API for socket and time.

Virtual filesystem benefits

Improve usability for clients. Clients can access the virtual filesystem as they want through intuitive ‘read’, ‘write’ and ‘append’ requests.

Improve usability for programs. Programs can use standard filesystem API and compile to WASI backend.

Virtual filesystem benefits, cont.

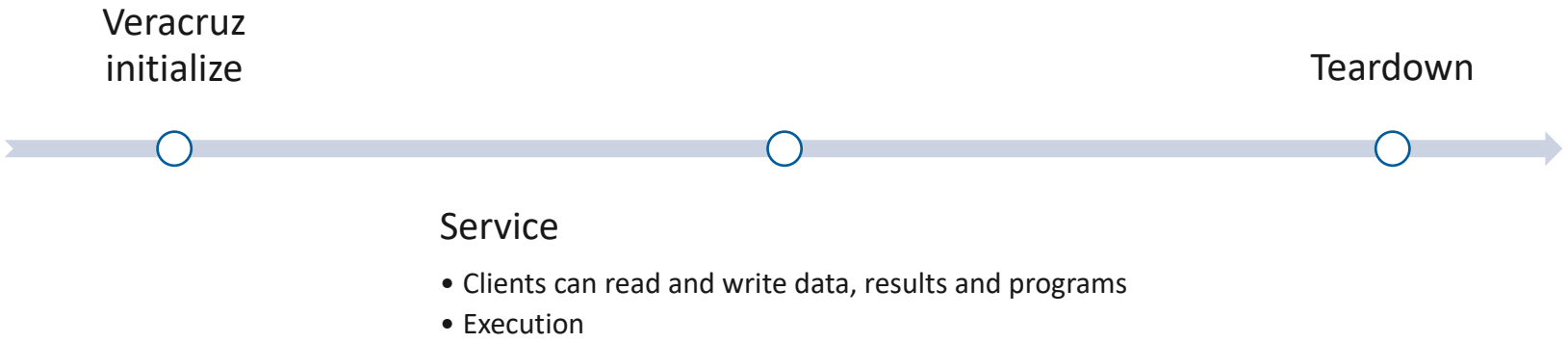
Reuse runtime. There is no limitation on how many programs can be stored in the Veracruz runtime. Clients can use the same runtime to achieve several tasks.

Observation #1: It is possible to pipe an output (file) of a program as an input (file) to another program (could be the same in the streaming case).

Observation #2: Do we want to allow a program directly call another program?

Reusable runtime

Veracruz runtime becomes a **service** (and the strict state machine is gone).



Work towards **streaming**.

Embedded clients (microcontroller-scale)

Why are we interested in embedded clients?

Embedded devices have limited compute:

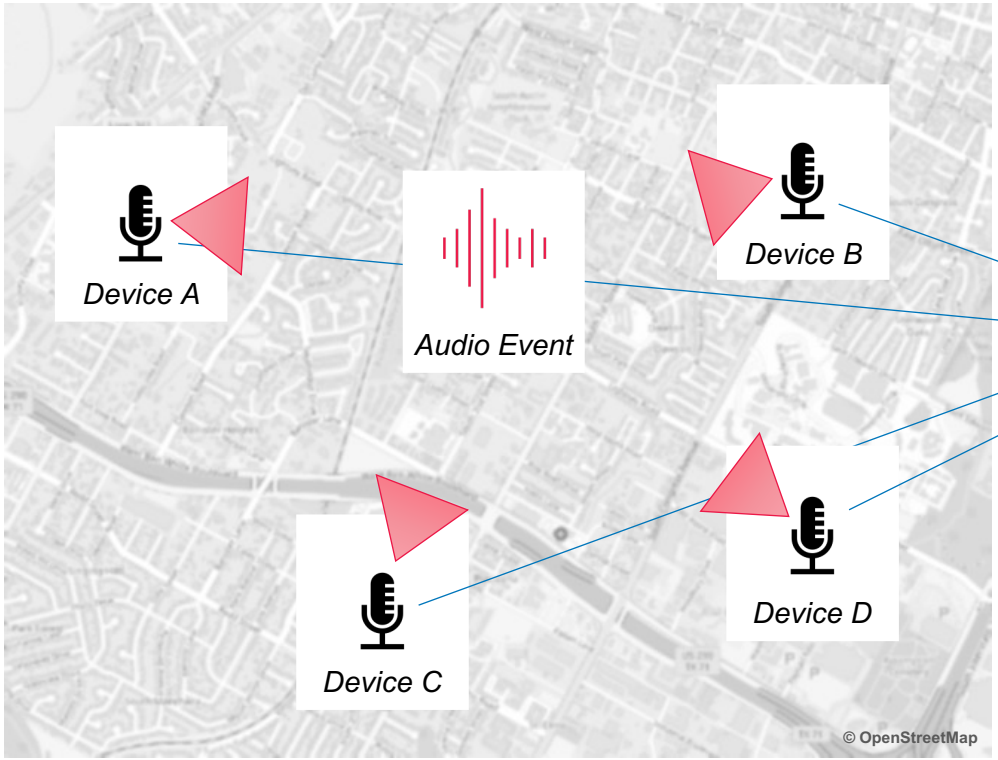
- Low-power
- Low-cost

Application complexity is increasing, requiring more compute. Offloading compute enables more complex applications without sacrificing power or price.

Low-latency edge compute further increases off-device compute options.

Offloading compute risks privacy.

Example use-case: Audio-event triangulation



Untrusted Edge Device

Event Location

Demo

CONFIDENTIAL COMPUTING CONSORTIUM

Conclusions

Briefly, some other things...

Some current work not yet discussed in this talk:

- Deploying Veracruz with Kubernetes,
- Veracruz as the basis for a FaaS infrastructure,
- Porting the Veracruz runtime as a Linux process to support new platforms,
- Spinning out a reusable hardware/attestation abstraction layer.

Future work

Two ideas we would like to pursue:

- Bounding the behaviour of the program binary,
- Larger, and multi-enclave use-cases for Veracruz.

Collaborators and contributors welcome!

We **welcome** new contributors to the project!

https://github.com/veracruz-project/veracruz

We have self-contained issues suitable for newcomers in our issue tracker

We also have a public Slack channel, `#veracruz`, under the CCC workspace

Closing remarks

Veracruz is an **exploration** of the consequences of widespread access to hardware support for Confidential Compute

How can these new features be used to design novel distributed systems?

With Veracruz, we we focussed on building a **framework**, which can be specialized to privacy-preserving distributed computations of interest

Use-cases for Veracruz are like those of **secure multi-party computations**

Yet, Veracruz is much easier to understand, easier to use, and easier to deploy, as this Webinar hopefully indicates

Questions?

Questions?

Questions?

CONFIDENTIAL COMPUTING
CONSORTIUM