golem

DEVCON 4

Solutions for trusted and private computations in Golem and in the wider Ecosystem

PIOTR JANIUK

golem

We need trustworthy computations

golem

Why do we need it?

01

Consensus in a trustless, decentralized network

02

Remote computation

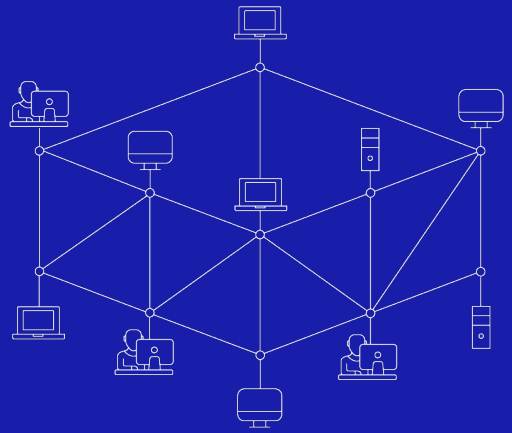
03

Decentralized services

04

And more...

In the context of Golem

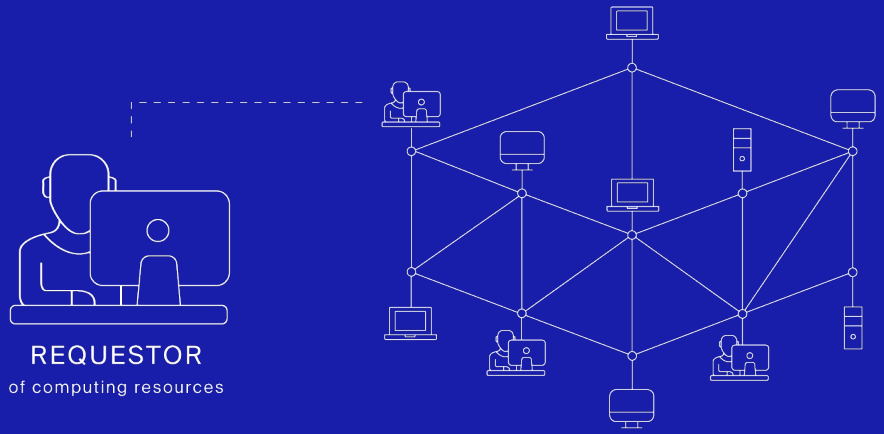


golem

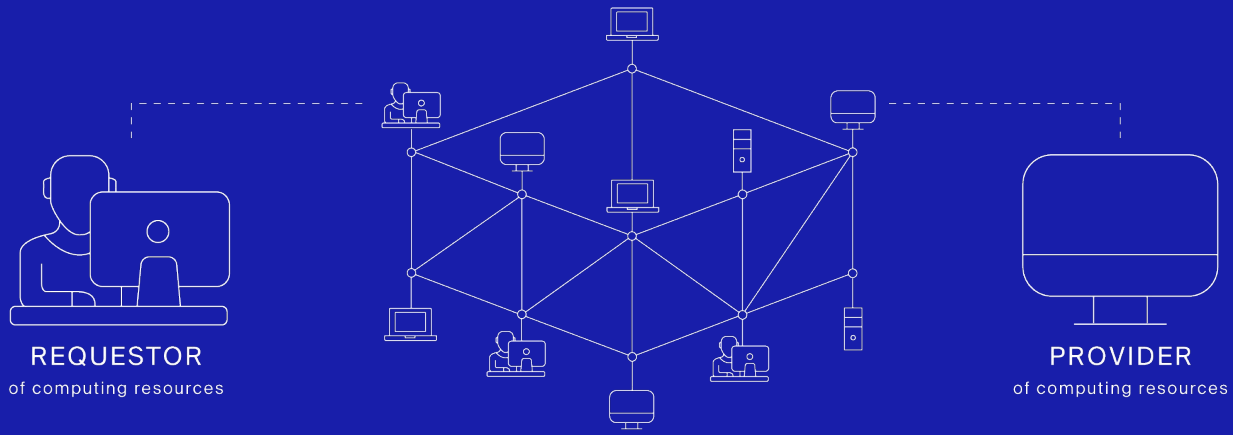
NETWORK

golem

In the context of Golem



In the context of Golem



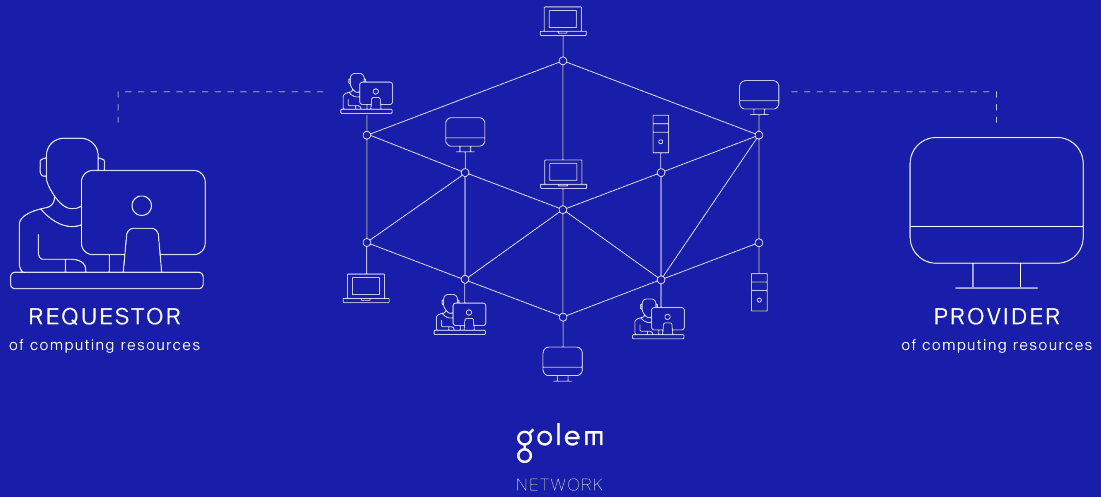
What is Golem?

01

A network of heterogeneous resources that can be either used by a **requestor** or provided to other participants by a **provider**.

02

Additional layers on top of the infrastructure (e.g., economy).

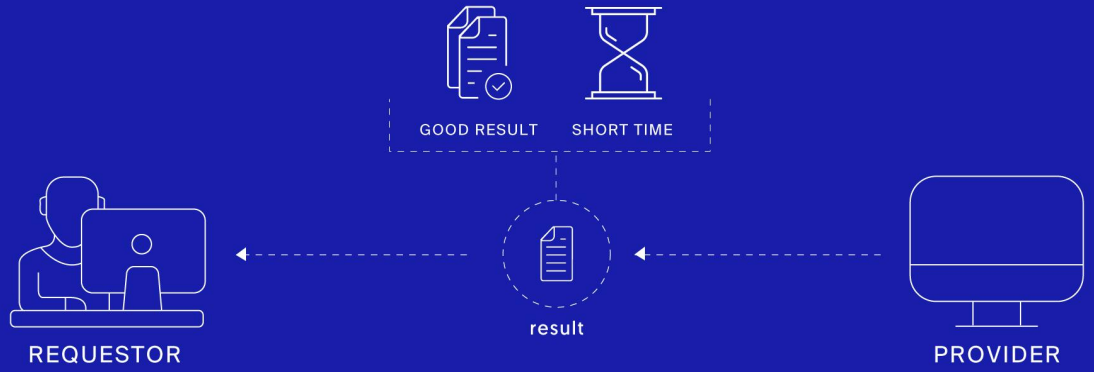


Problem statement

01

Requestor wants to be sure that

- He receives valid results from the provider after a trustworthy execution on the provider's machine
- Gets results within a reasonable time

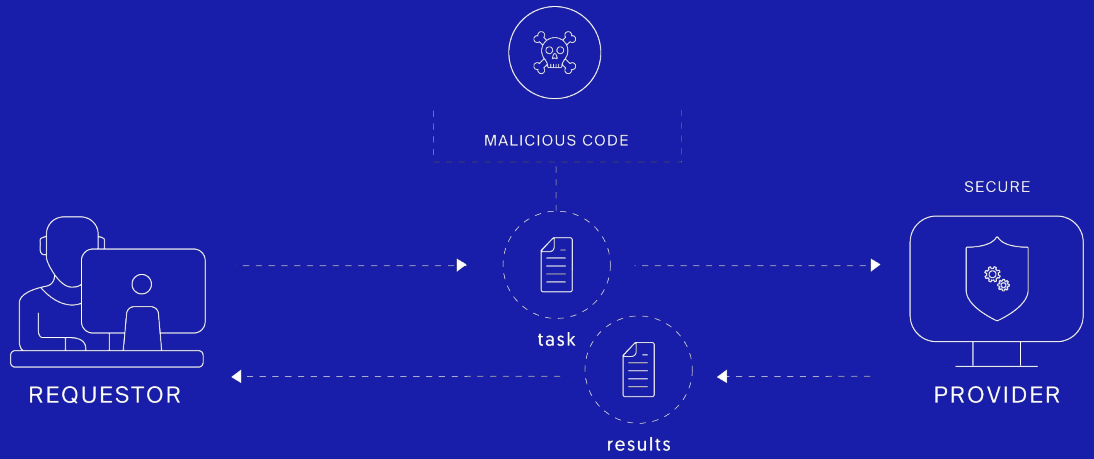


Problem statement

02

Provider on the other hand

- Needs to be protected from malicious tasks



High-level requirements

01

Requestor should be able to run an arbitrary, unmodified binary efficiently

High-level requirements

02

Requestor should be able to run an arbitrary, unmodified binary efficiently

Task input data has to be the same as prepared by the requestor

High-level requirements

03

Requestor should be able to run an arbitrary, unmodified binary efficiently

Task input data has to be the same as prepared by the requestor

The binary run by a provider must be the one requestor wanted

High-level requirements

04

Requestor should be able to run an arbitrary, unmodified binary efficiently

Task input data has to be the same as prepared by the requestor

The binary run by a provider must be the one requestor wanted

The execution of the task has to be carried out without tampering

High-level requirements

05

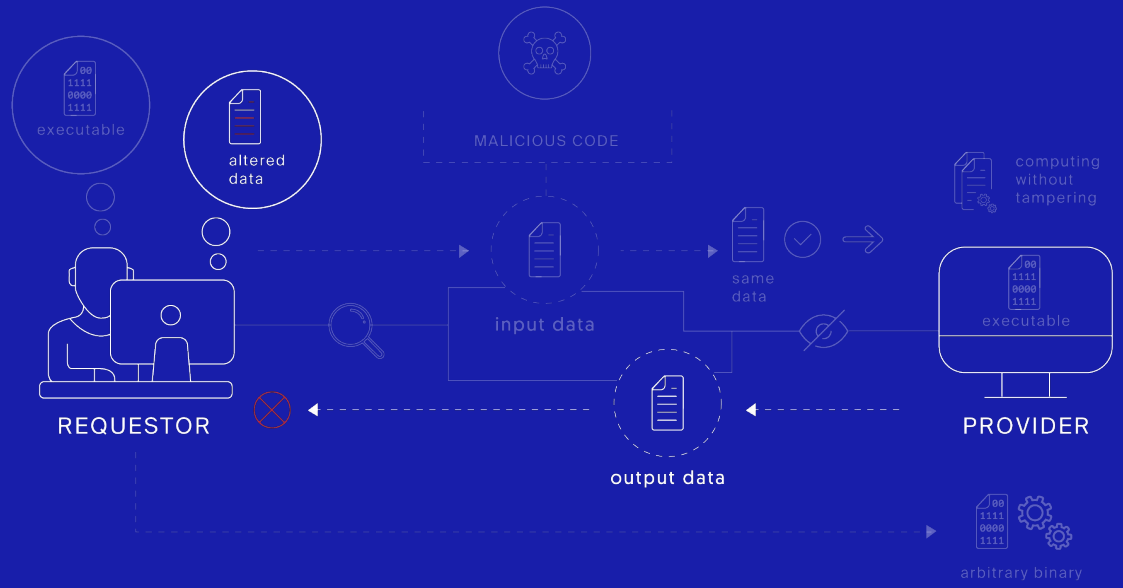
Requestor should be able to run an arbitrary, unmodified binary efficiently

Task input data has to be the same as prepared by the requestor

The binary run by a provider must be the one requestor wanted

The execution of the task has to be carried out without tampering

Output data cannot be altered without detection



High-level requirements

06

Requestor should be able to run an arbitrary, unmodified binary efficiently

Task input data has to be the same as prepared by the requestor

The binary run by a provider must be the one requestor wanted

The execution of the task has to be carried out without tampering

Output data cannot be altered without detection

Only the requestor can see the plain text input and output data

High-level requirements

06

Requestor should be able to run an arbitrary, unmodified binary efficiently

Task input data has to be the same as prepared by the requestor

The binary run by a provider must be the one requestor wanted

The execution of the task has to be carried out without tampering

Output data cannot be altered without detection

Only the requestor can see the plain text input and output data

Provider has to be protected from malicious binaries

golem

Meeting the requirements

01

Limiting the class of problems

- **deterministic (PoW or verification)**
- **nondeterministic (probabilistic verification)**

02

Introducing external sources of trust

03

By means of the infrastructure (TEEs)

04

Hybrid approach

Choosing the best approach

01

Tradeoff

- Inherent to the method
- Task specific
- Runtime dependent

02

Focus on a single one

- Based on the hardware infrastructure
- Allows remote computations

golem

Meeting the requirements

SGX, Graphene and Graphene-ng

SGX - short recap

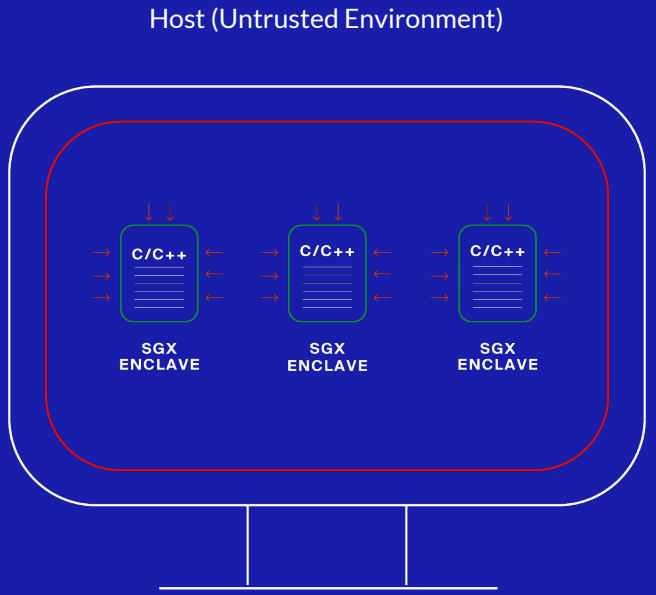
01

Overview

- *An Intel® architecture extension designed to increase the security of application code and data*
- Enclave based computation model (application is run in containers called enclaves, which are supposed to be protected from the host)
- Provides a mechanism for remote attestation

Developer perspective - an enclave

- Static interface specification
- Source code + Intel SDK
- By default limited to computations

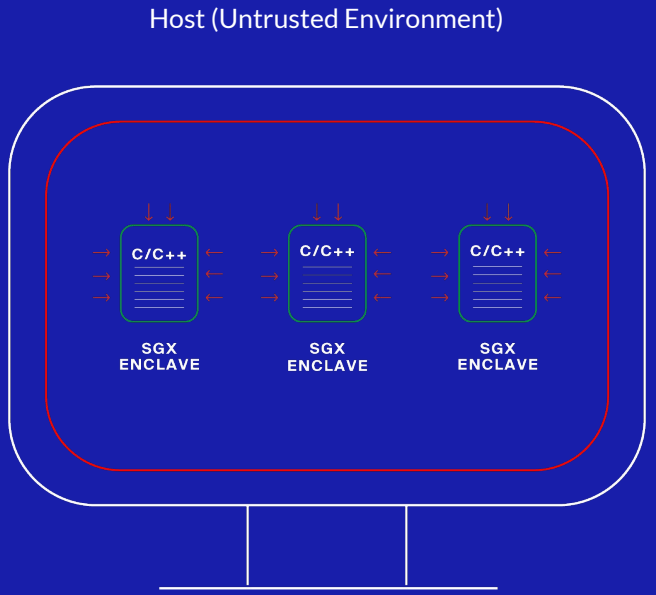


SGX - short recap

02

Summary

- Provides important security features
- Does not allow to run arbitrary applications in the enclave
- May be treated as a building block for more generic solutions



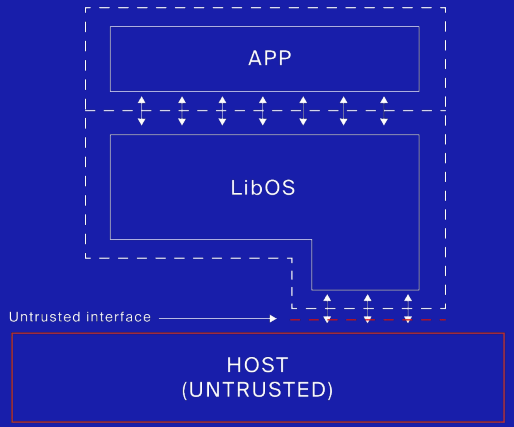
Graphene overview

01

Features

- **A framework with LibOS, which simulates the OS for the app in the enclave**
- **Runs unmodified, full Linux apps protected from the host**
- **Makes use of Intel SGX**

Graphene architecture



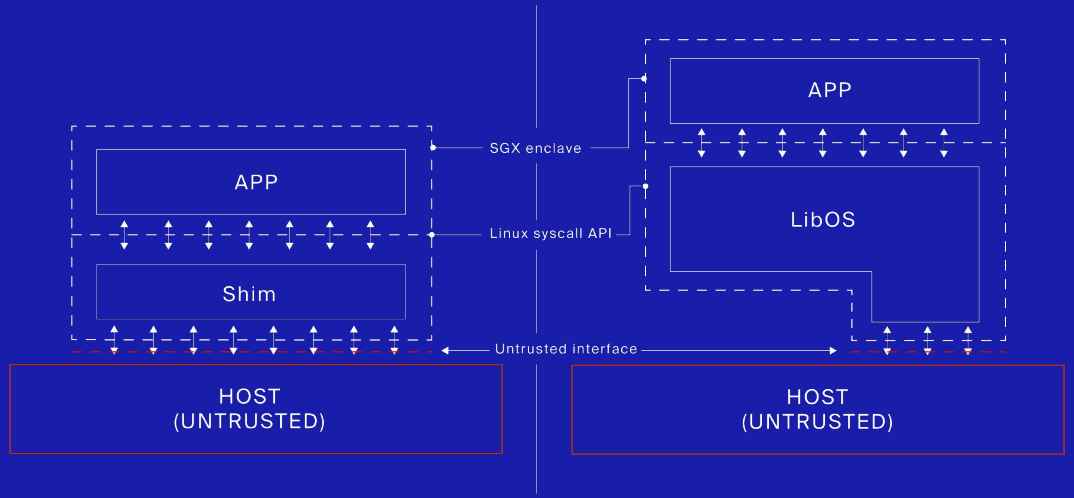
Graphene overview

02

Graphene vs other approaches

- TCB vs attack surface
- Flexibility - other host or guest OS can be supported by changing a single component
- Better suited to sandboxing

Graphene architecture

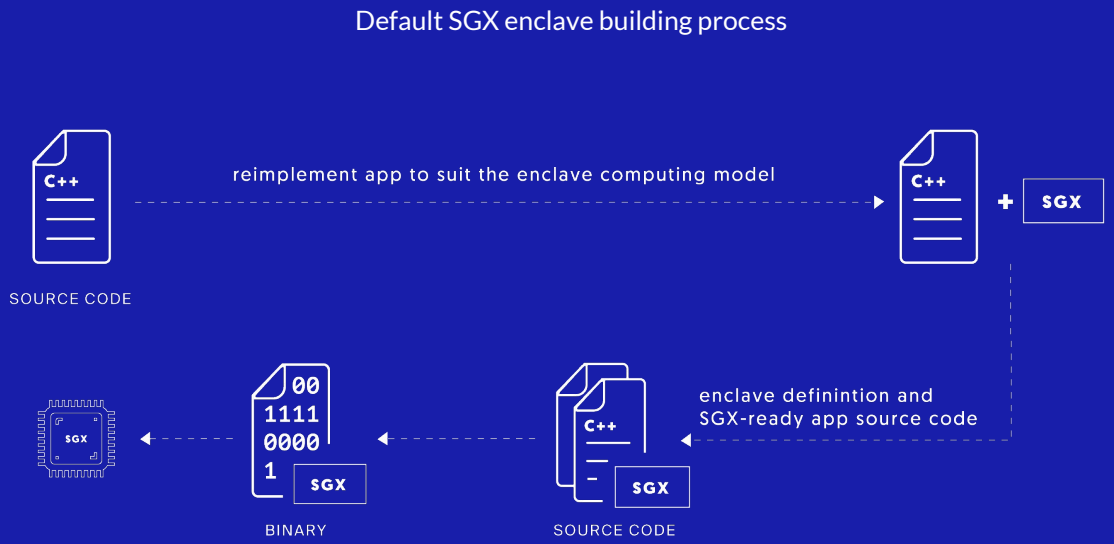


Graphene overview

03

With Graphene, a developer doesn't have to

- **Rewrite the code to suit SGX needs**
- **Specify the static enclave interface**
- **Compile the enclave**

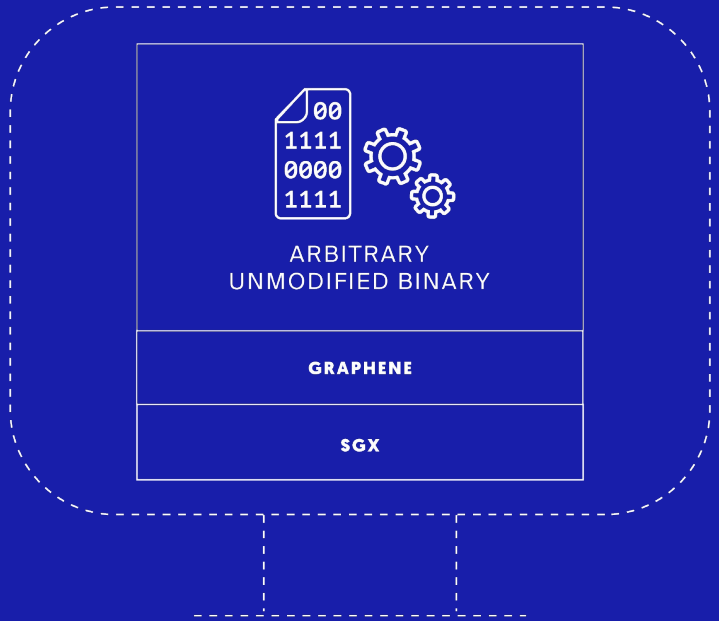


Graphene overview

04

With Graphene:

- An arbitrary, unmodified Linux binary can be run inside an enclave



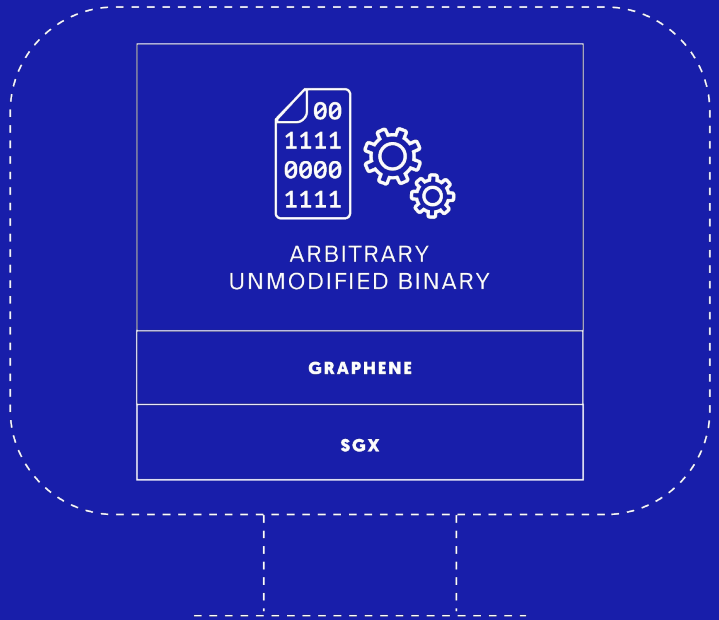
Graphene overview

05

With Graphene:

- An arbitrary, unmodified Linux binary can be run inside an enclave

\+ The process still requires manual work to configure, deploy and run the application



The next step

Graphene-ng

The next step

Graphene-ng = Graphene

+ Protected files
+ Docker support
+ Tools (scripts)
+ Bug fixes

The next step

Graphene-ng = Graphene

Resulting in a better UX regarding interaction with the enclave for both, the developer and the end user.

+ Protected files
+ Docker support
+ Tools (scripts)
+ Bug fixes

Graphene-ng by example

PoC integration with Golem

Provider setup

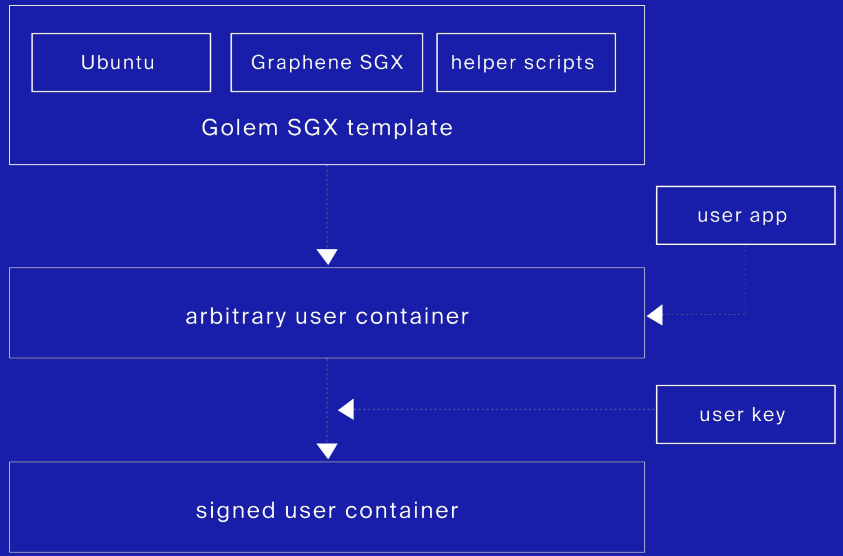
02

SGX must be enabled on provider's machine

Protected Files configuration:

- **Enclave manifest**
- **Running the docker image**

Graphene-ng Docker configuration



Provider setup

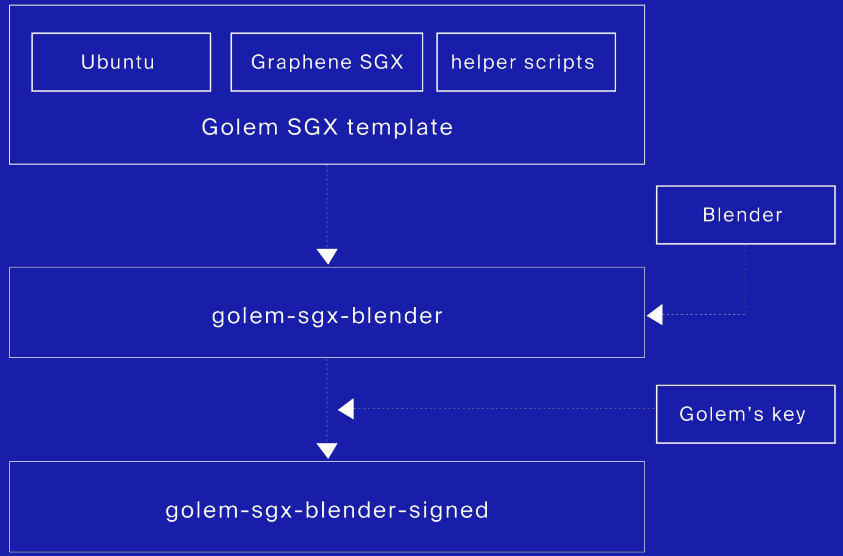
03

SGX must be enabled on provider's machine

Protected Files configuration:

- **Enclave manifest**
- **Running the docker image**

Graphene-ng Docker for Brass Golem configuration

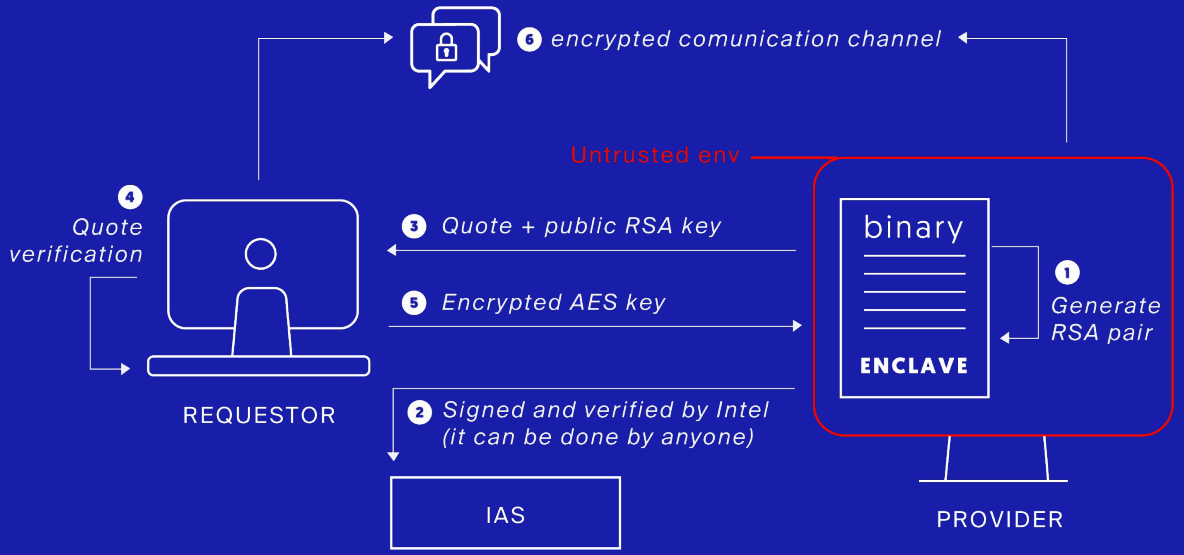


Initialization - setup and handshake

01

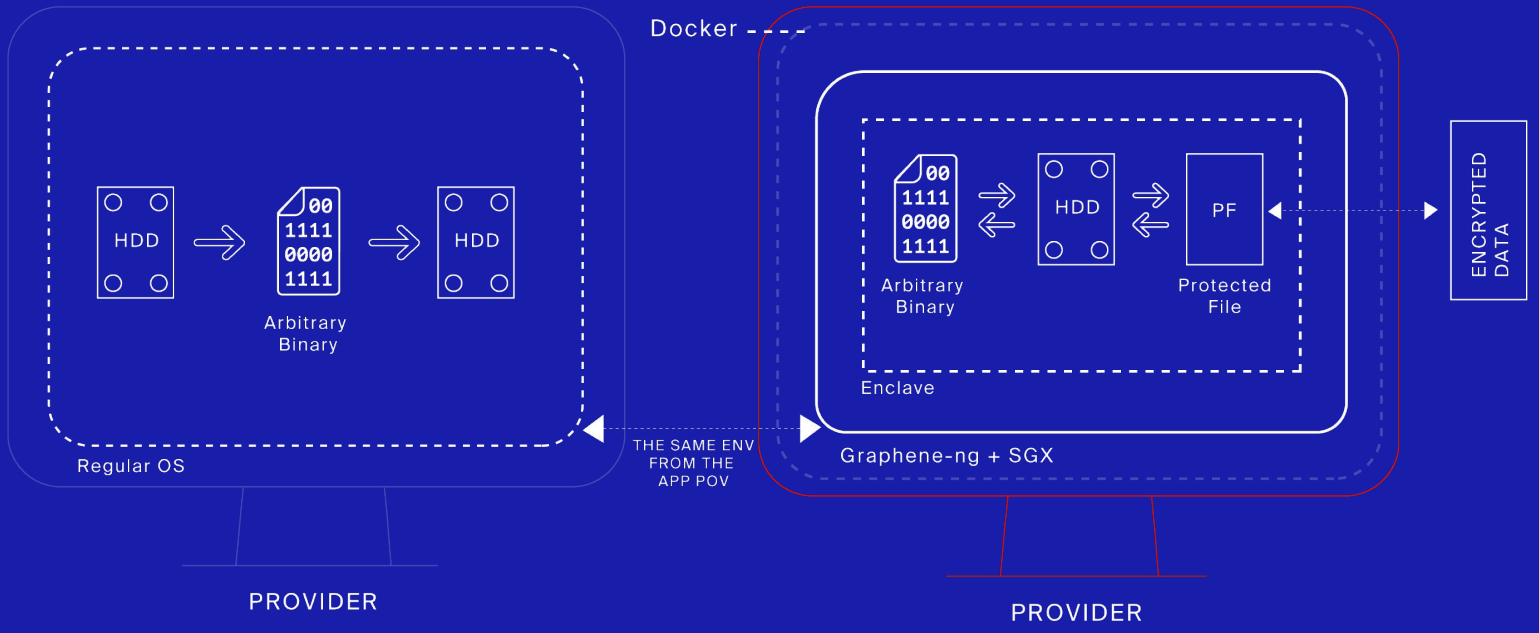
The process

- **Semi-automatic**
- **Uses convenience scripts**



golem

Application POV

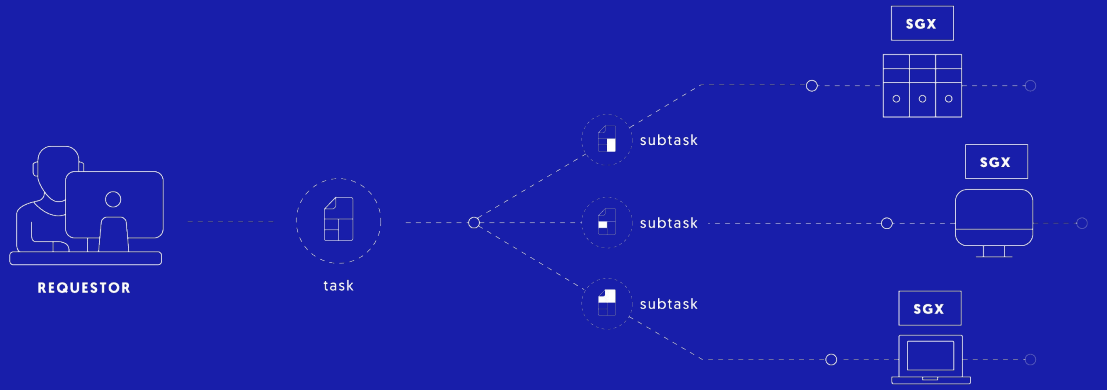


golem

Requestor POV

01

Requestor looks for SGX nodes in a P2P network



golem

Requestor POV

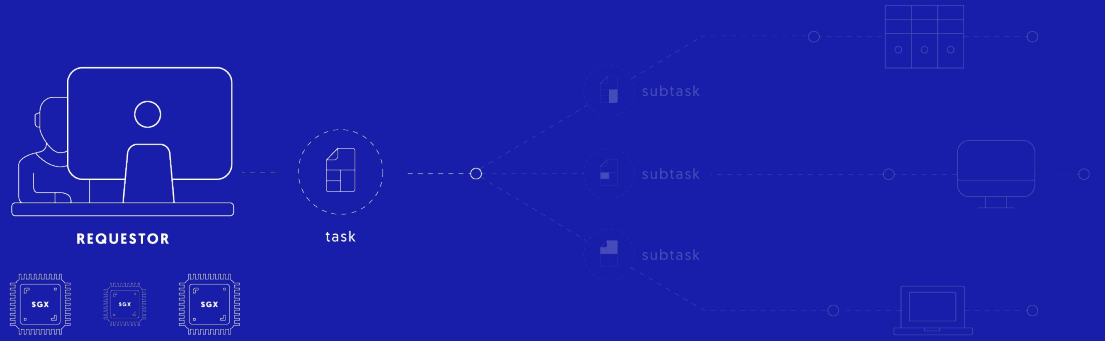
02

And connects to available nodes that offer the SGX architecture

Requestor POV

03

It can be envisioned as if the local machine had more compute resources available locally

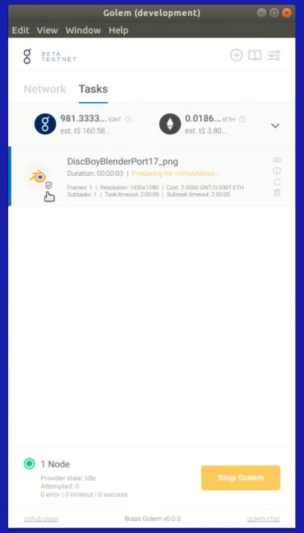


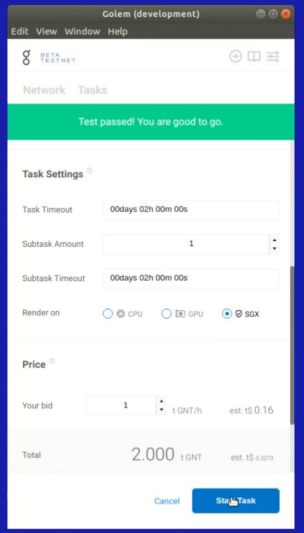
And yes, it works

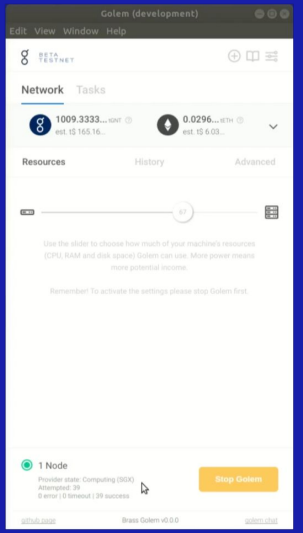
01

Benefits for Golem and the community

- Confidential remote computation
- Using unmodified binaries
- SGX as an infrastructure in Golem







golem

Next steps

01

SGX:

- Liberation: IAS, FLC
- Mitigation of known attacks
- EPC size and efficiency

02

Development:

- Stable and efficient Graphene-ng
- Proofs of computation
- Windows support

golem

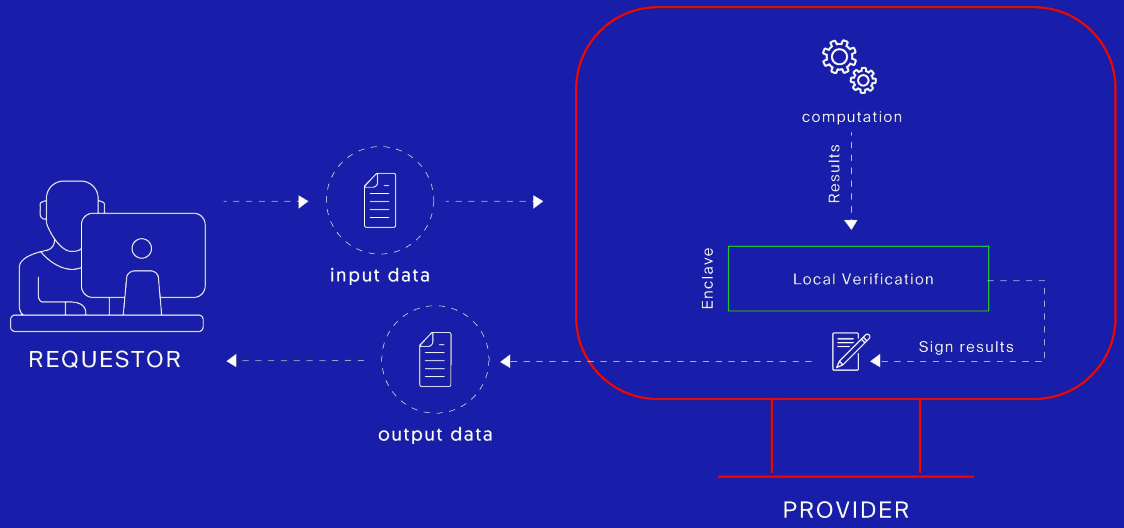
Graphene-ng Use Cases

01. Golem ecosystem

Local Verification

01

- Task is computed by an untrusted host
- And verified in an enclave run by the provider



Golem Unlimited

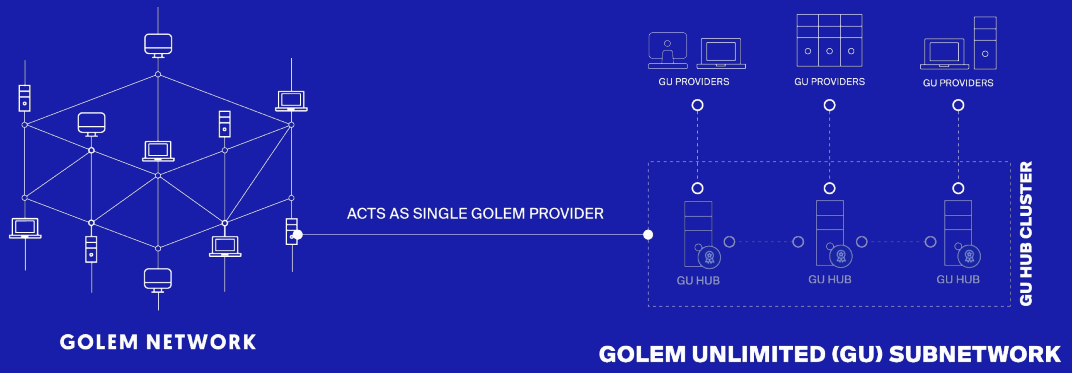
01

Golem Unlimited

- Infrastructure based on a LAN-like network setup

Two trusted actors:

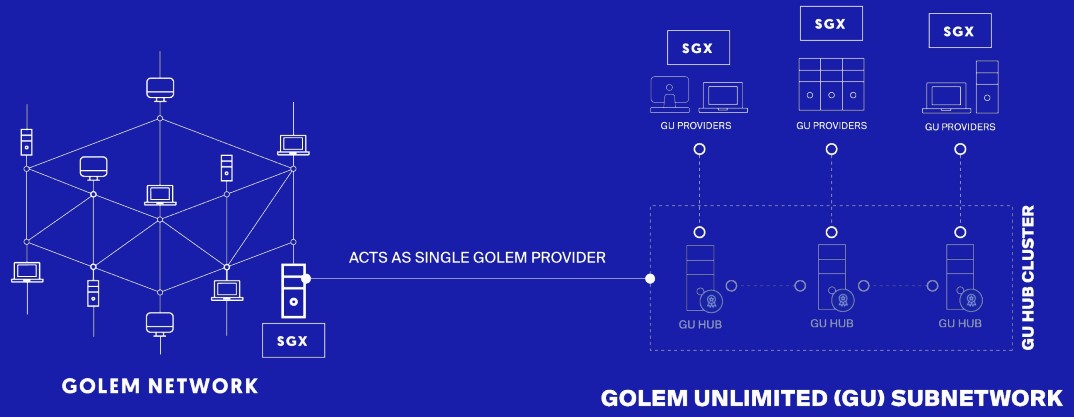
- The hub - requestor role
- Provider nodes



Golem Unlimited - powerful component with SGX

02

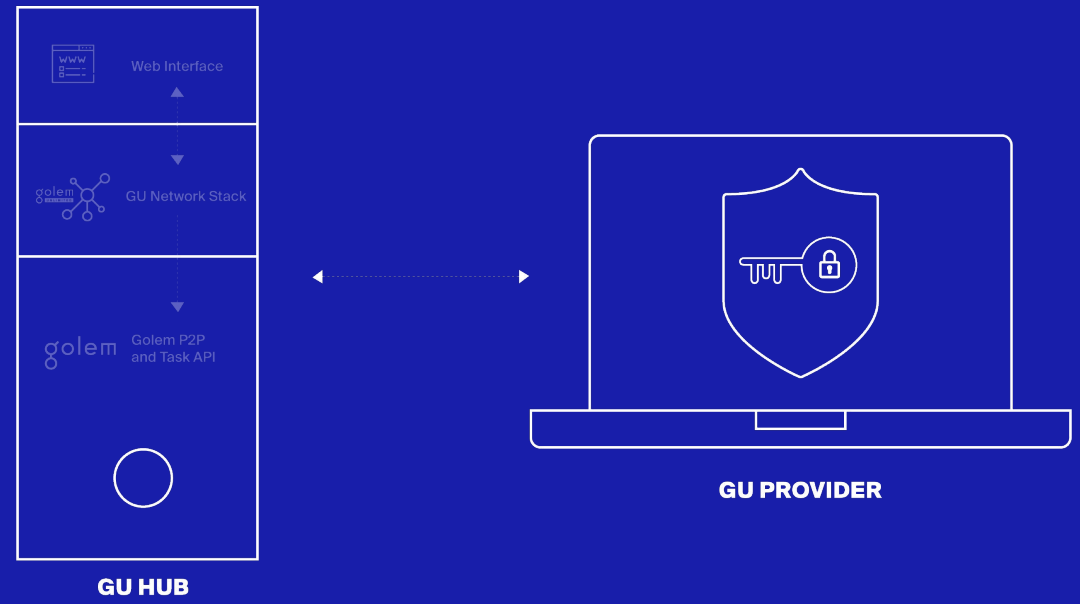
- Each Unlimited provider exposes SGX backend
- Can be envisioned as a powerful component with SGX added to the Golem network



Golem Unlimited - storing identities

03

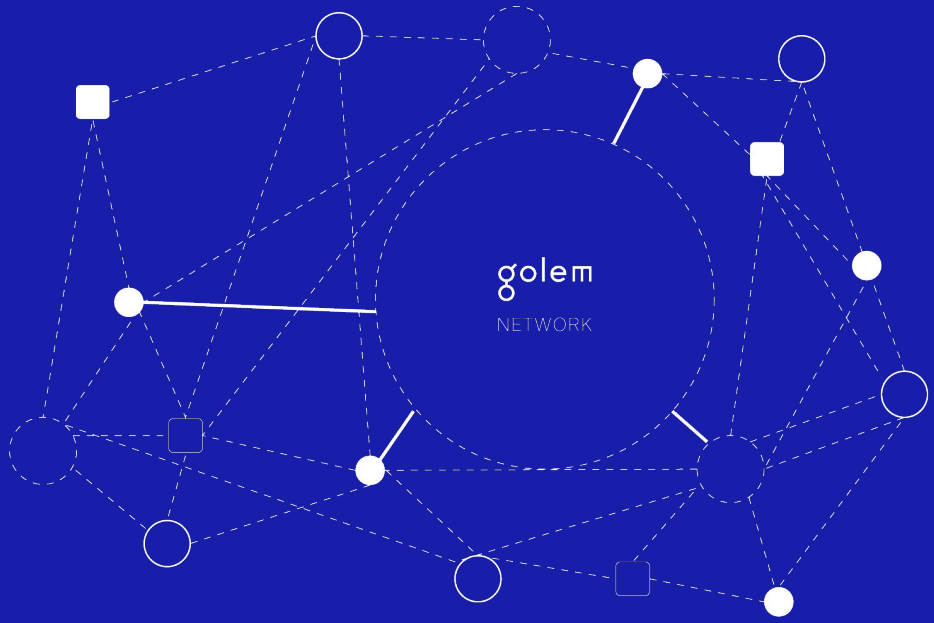
- Storing the identity information only inside enclaves
- Binds identities to physical machines in a passwordless manner



Use Cases - other Golem integrations

01

Integrations with Golem by external developers



golem

Graphene-ng Use Cases

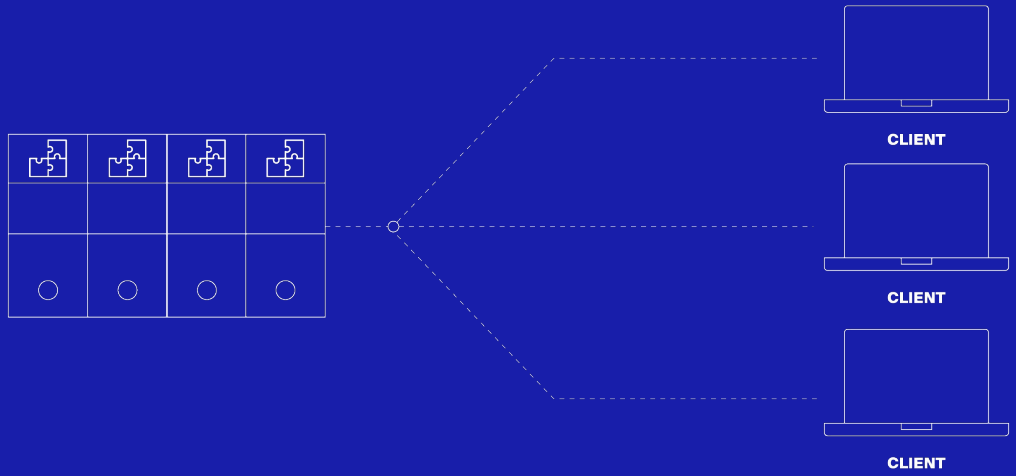
02. Wider audience

golem

Decentralized server implementation

01

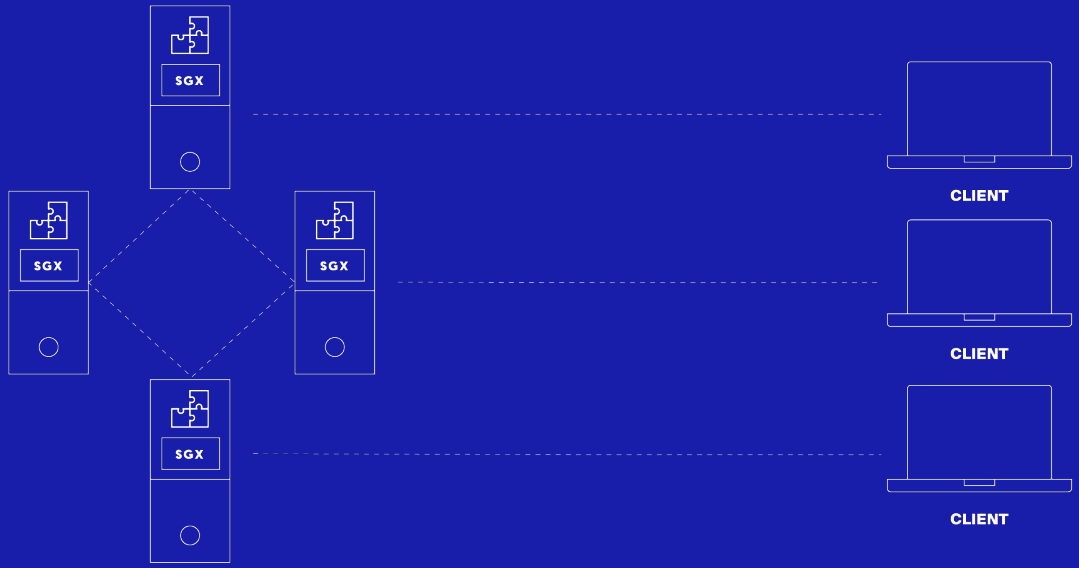
- Centralized infrastructure



Decentralized server implementation

02

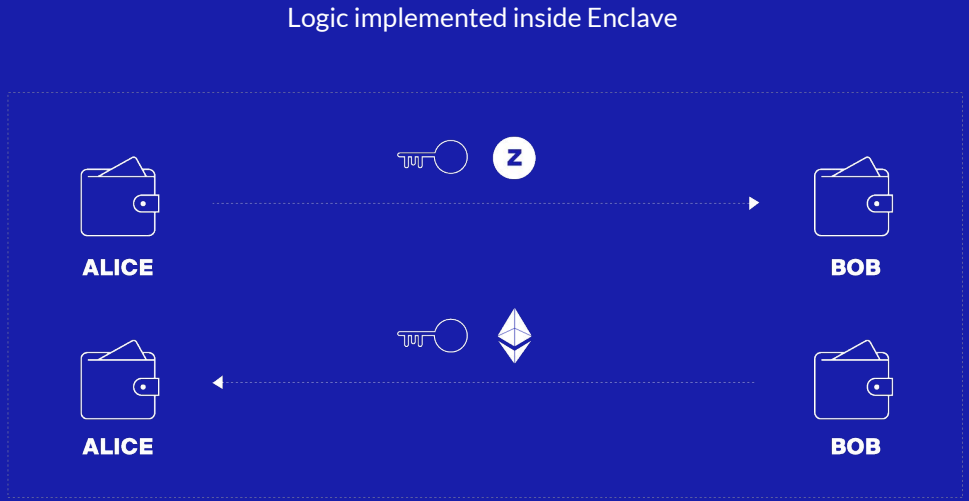
- Decentralized architecture
- May be potentially used to facilitate secure multi-party computation



Atomic swap

01

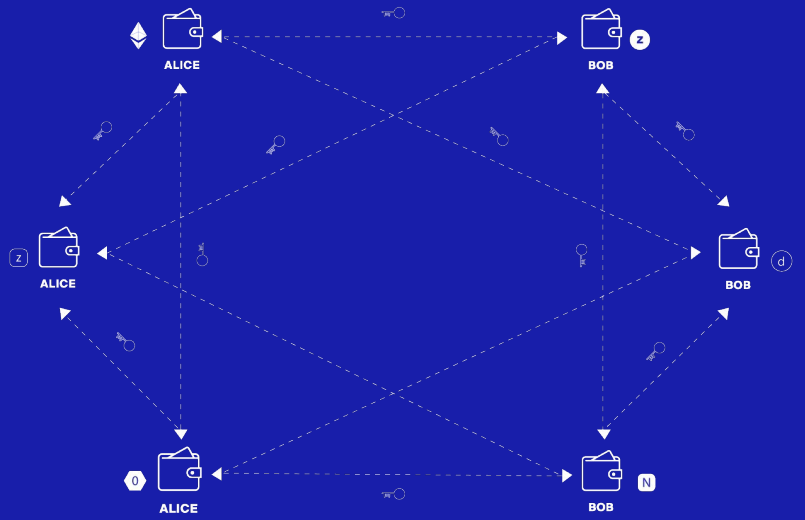
Cross-blockchain atomic swap



Distributed exchanges

02

Implemented by means of cross-blockchain atomic swap



golem

Storing secrets

01

- Sealing private keys locally
  - Hardware wallet primitive
- Sealing private keys remotely

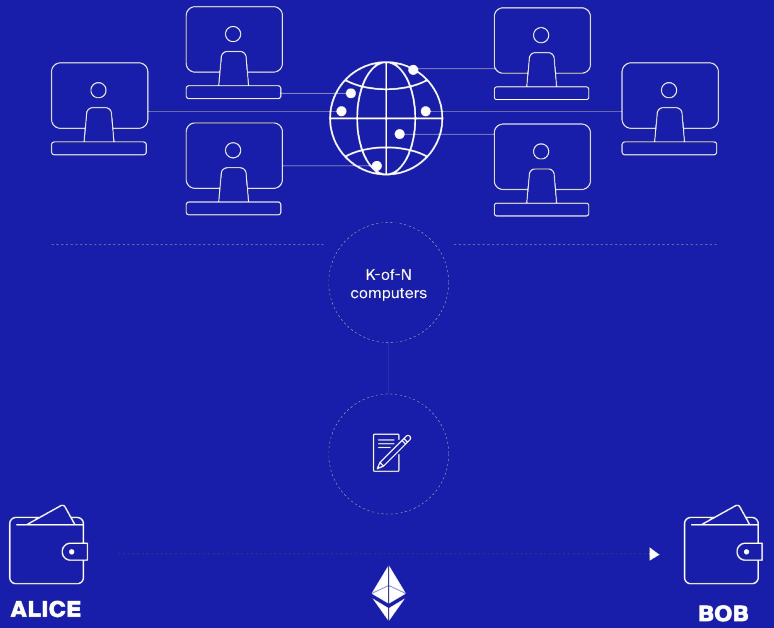
STORING PRIVATE KEYS IN CLOUD

golem

Storing secrets

02

- Threshold signatures
- Identity stored in a decentralized way



golem

Graphene-ng Use Cases

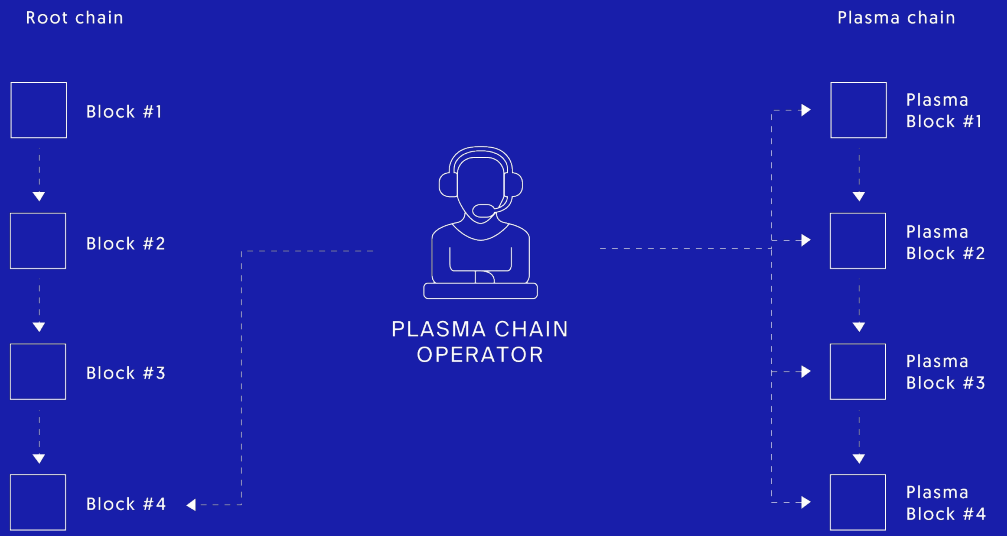
03. In existing projects

golem

Minimal Viable Plasma

01

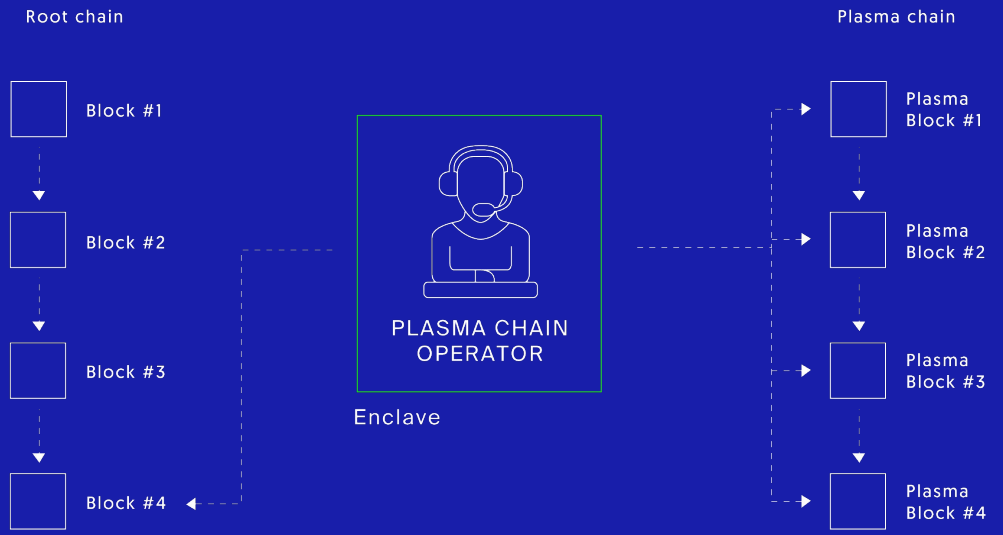
Minimal Viable Plasma with operator



Minimal Viable Plasma

02

Plasma chain operator can be implemented in an SGX enclave



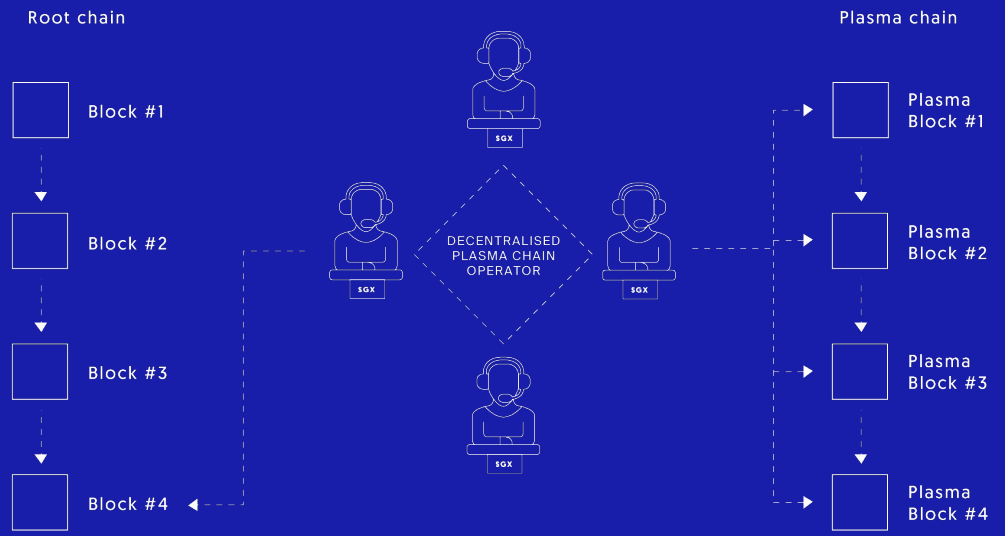
golem

Minimal Viable Plasma

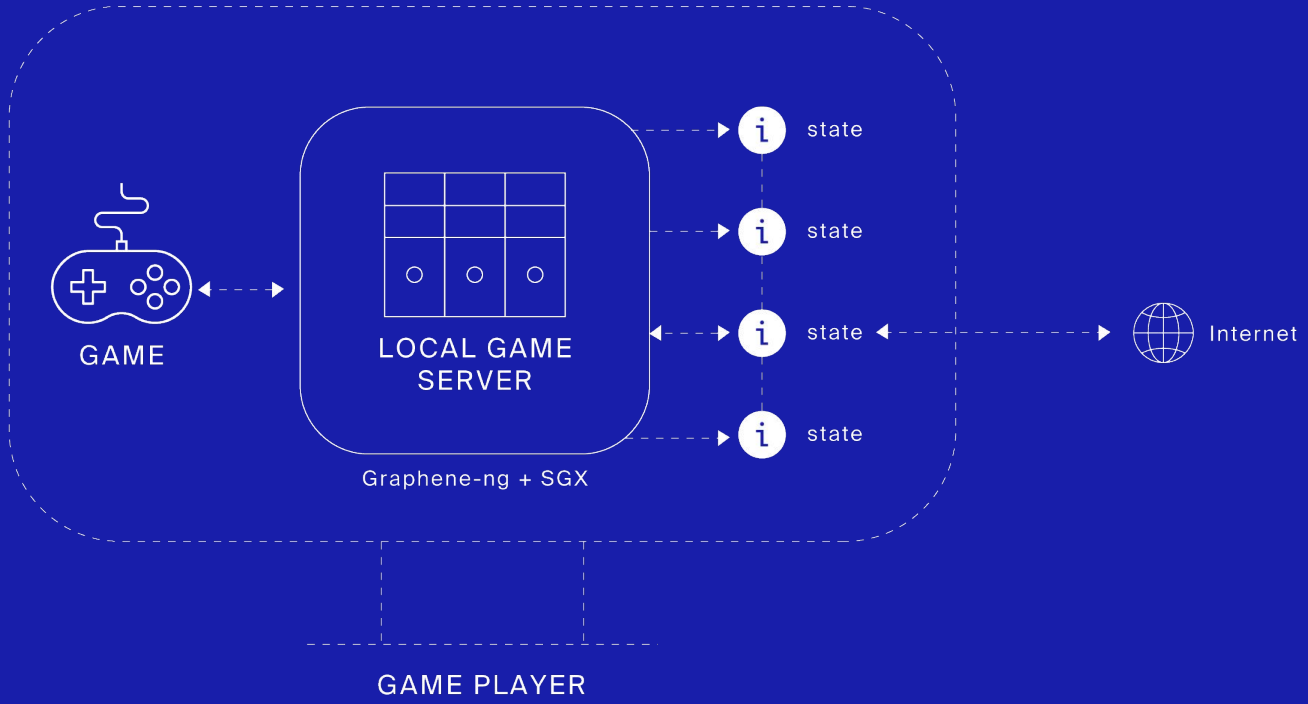
03

Plasma chain operator can be implemented in an SGX enclave

And additionally decentralised

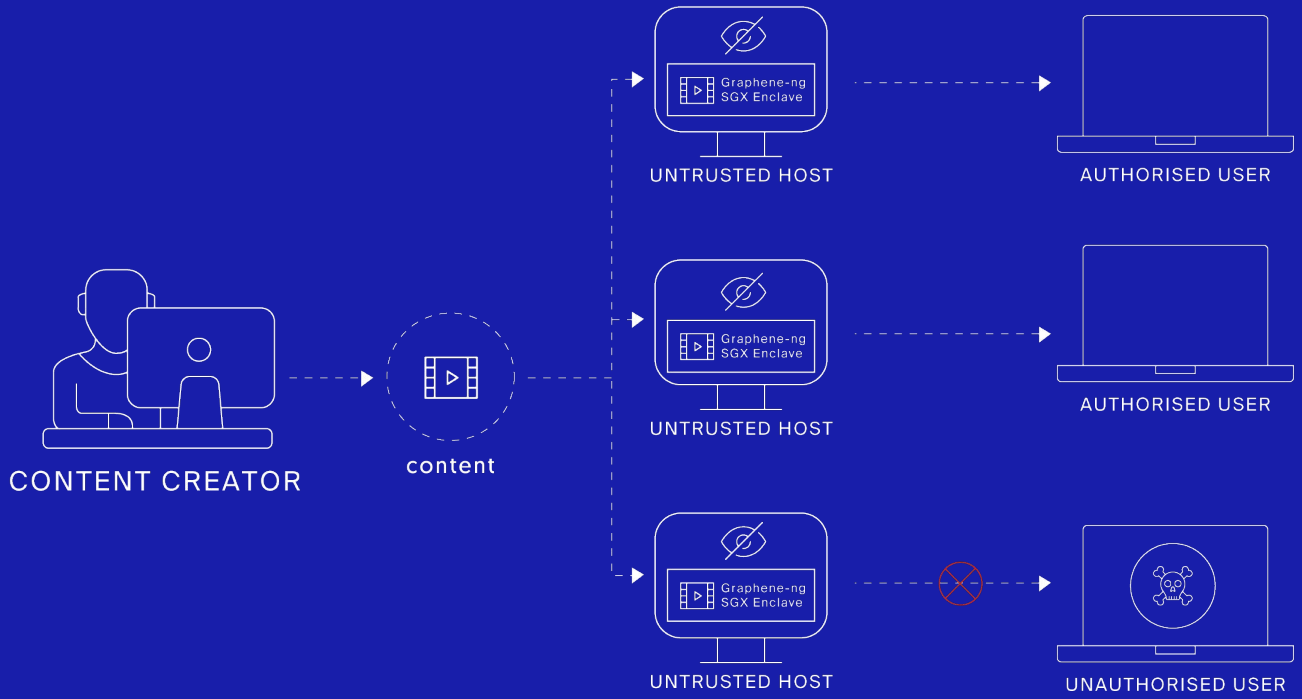


Hoard

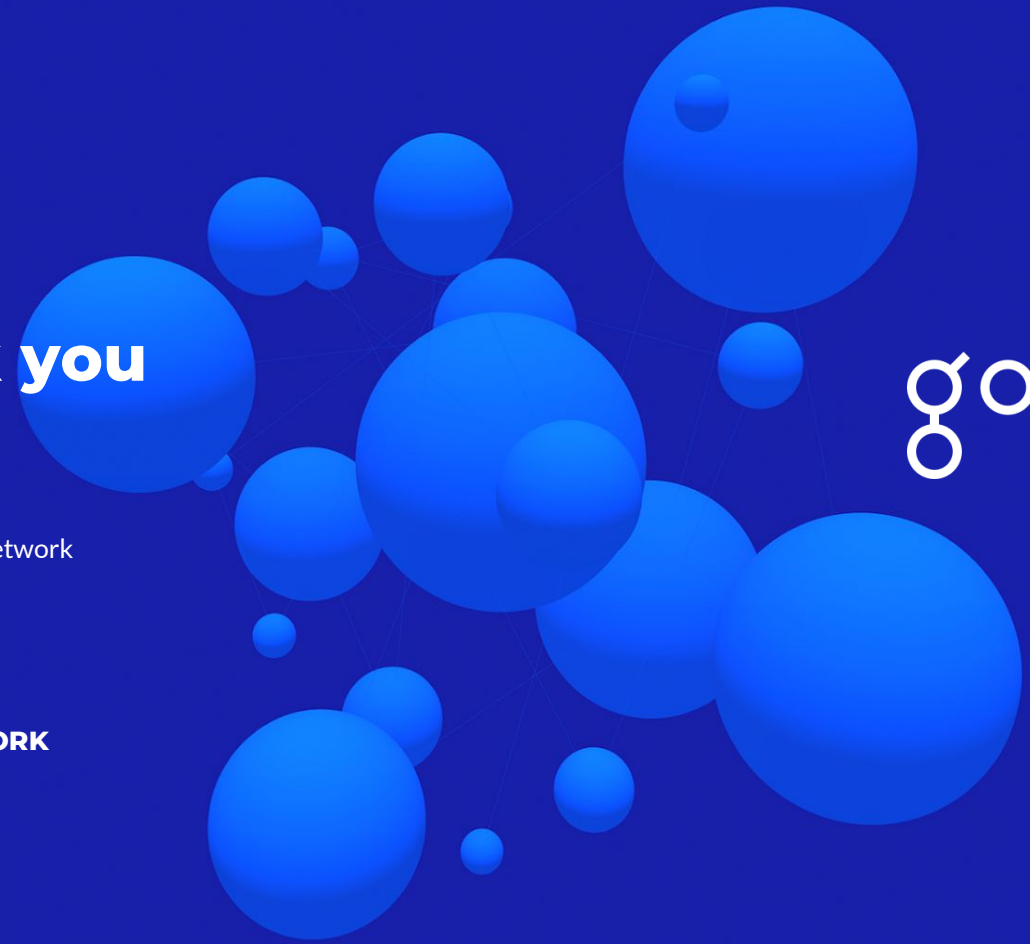

GAME
LOCAL GAME SERVER
Graphene-ng + SGX
state
state
state
state
Internet
GAME PLAYER

golem

Data streaming



Thank you



golem

PIOTR JANIUK

contact@golem.network

GOLEM.NETWORK