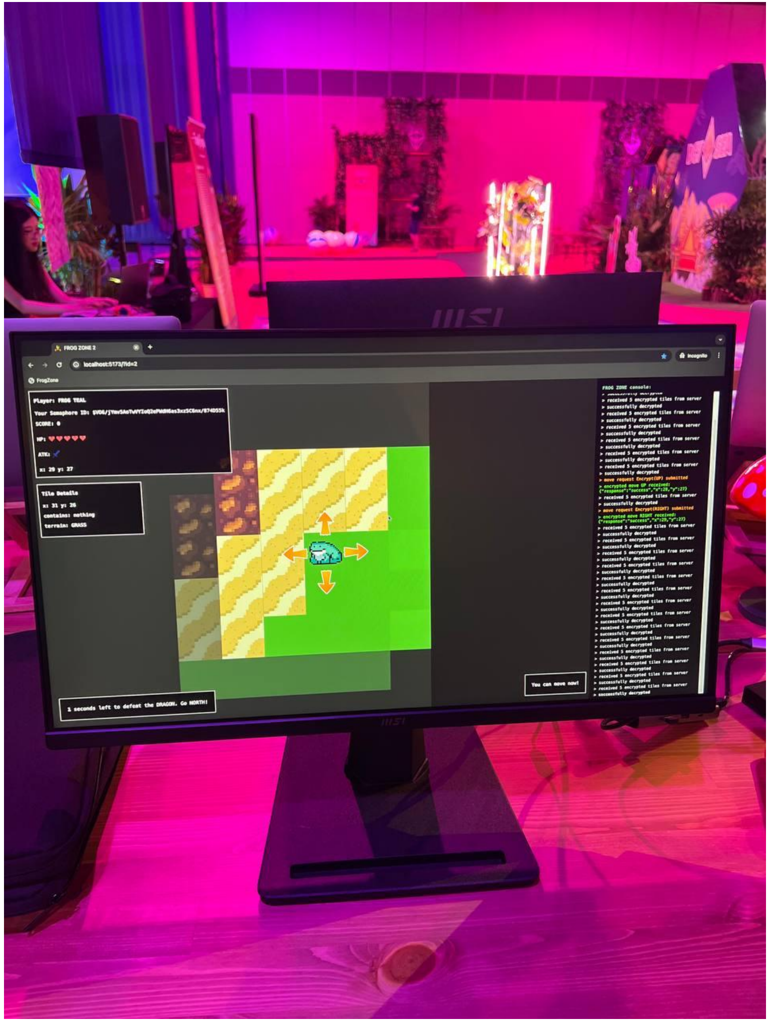
How to Hallucinate a Server

gubsheep - Devcon SEA

INTRO

Gubsheep

ENTER THE FROGZONE


FROG ZONE 2
localhost:5173/?id=2
Incognito
FrogZone
Player: FROG TEAL
SCORE: 0
HP:
ATK:
x: 29 y: 27
Tile Details
x: 31 y: 26
contains: nothing
terrain: GRASS
FROG ZONE console:
received 5 encrypted tiles from server
successfully decrypted
move request Encrypt(UP) submitted
move request Encrypt(RIGHT) submitted
You can move now!
1 seconds left to defeat the DRAGON. Go NORTH!
MSI


MSI

our Semaphore ID:
/O6/jYmv5AoTwVYIoQ2ePWdH6as3xz5C6nx/874D55k

CORE: 10

P:

TK:

: 4 y: 23

ile Details

: 5 y: 23

ontains: monster

errain: GRASS

onster Details

ame: IMP

escription: A cute little guy just having a
ad day

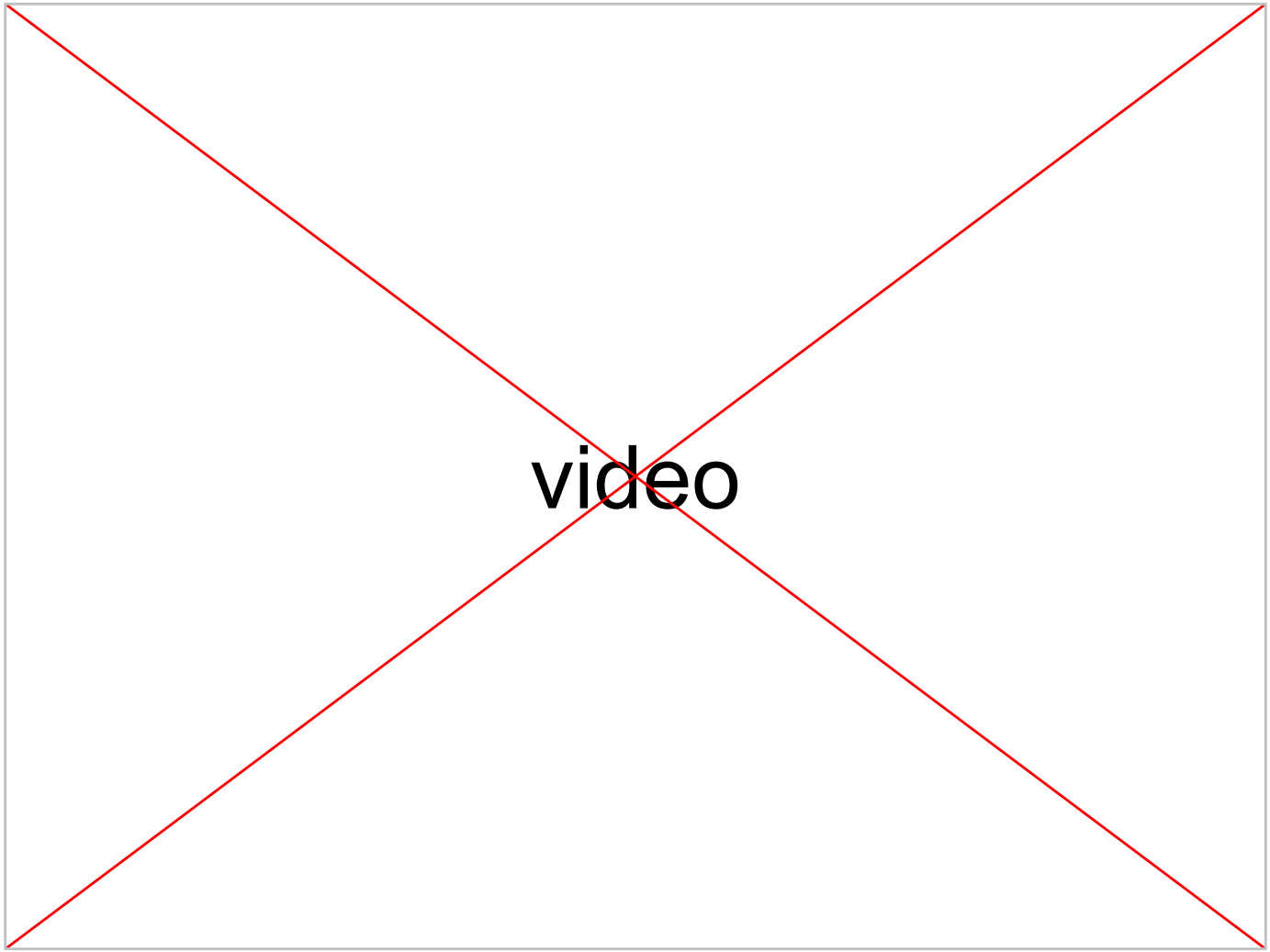
P:

TK:

256 seconds left to defeat the DRAGON. Go NORTH!

Next move available in: 2.4 seconds

> successfully decrypted
> received 5 encrypted tiles from server
> successfully decrypted
> move request Encrypt(UP) submitted
> encrypted move UP received:
{"response":"success","x":3,"y":24}
> received 5 encrypted tiles from server
> successfully decrypted
> received 5 encrypted tiles from server
> successfully decrypted
> received 5 encrypted tiles from server
> successfully decrypted
> move request Encrypt(LEFT) submitted
> received 5 encrypted tiles from server
> successfully decrypted
> encrypted move LEFT received:
{"response":"success","x":3,"y":24}
> received 5 encrypted tiles from server
> successfully decrypted
> received 5 encrypted tiles from server
> successfully decrypted
> move request Encrypt(RIGHT) submitted
> received 5 encrypted tiles from server
> successfully decrypted
> encrypted move RIGHT received:
{"response":"failure"}
> move request Encrypt(RIGHT) submitted
> encrypted move RIGHT received:
{"response":"success","x":4,"y":24}
> received 5 encrypted tiles from server
> successfully decrypted
> received 5 encrypted tiles from server
> successfully decrypted
> received 5 encrypted tiles from server
> successfully decrypted
> received 5 encrypted tiles from server
> successfully decrypted
> move request Encrypt(UP) submitted
> received 5 encrypted tiles from server
> successfully decrypted
> encrypted move UP received:
{"response":"success","x":4,"y":23}
> received 5 encrypted tiles from server
> successfully decrypted
> received 5 encrypted tiles from server
> successfully decrypted
> received 5 encrypted tiles from server
> successfully decrypted
> move request Encrypt(RIGHT) submitted
> encrypted move RIGHT received:


video

Gameplay

- Four frogs on a 32x32 grid
- Frogs have HP and ATK
- You can move up, down, left, or right
- There are items you can find and pick up to get more HP and ATK
- There are monsters moving around the map that you can fight and kill
- At the top there is a dragon you can fight. If you beat him you win the game
- You can only see in the 5x5 area around you
- Players can move once every ~5 seconds.

The game in numbers (part 1)

The total size of the game state is about 150 bytes (0.15kb).

The frontend is React + Phaser. The source code for the backend is about 500 lines of C++.

Took a month for a team of devs from Gauss + 0xPARC + PSE to build together.

What is going on here?

The backend of this game is running **in fully homomorphic encryption.**

To our knowledge, this is the first multi-user application with a persistent backend that is running inside of FHE.

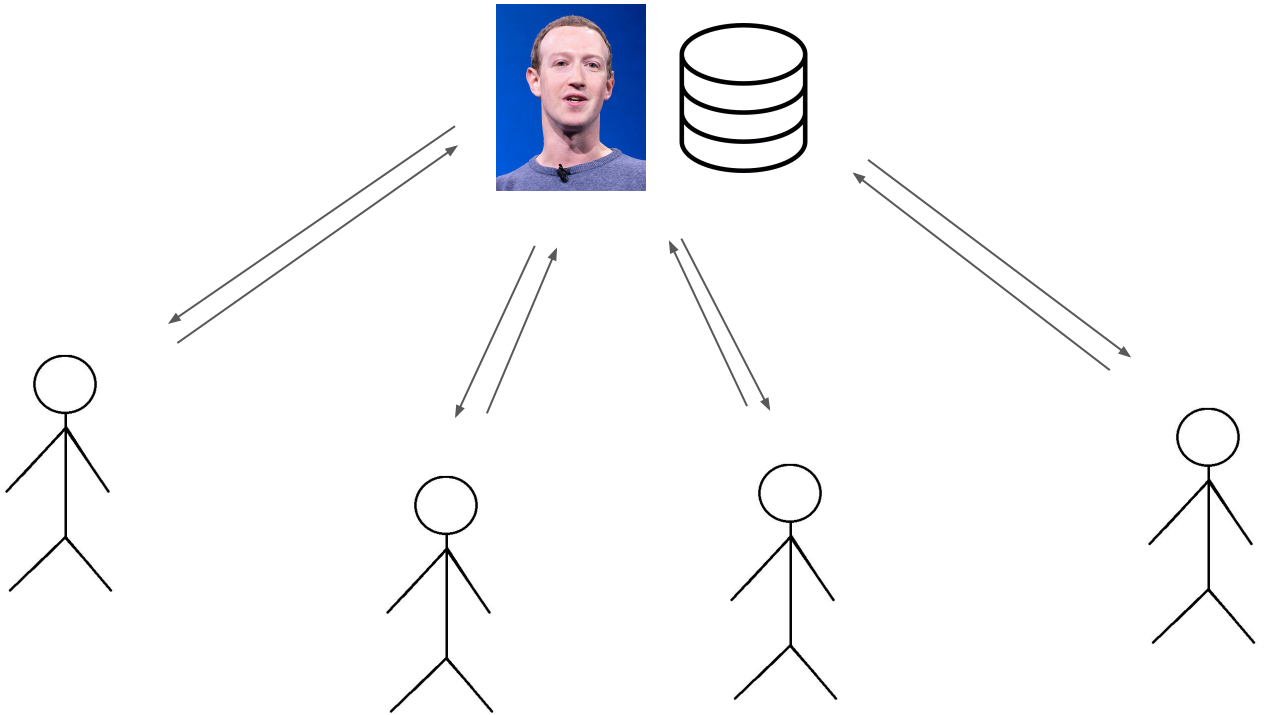
All of the state is encrypted; all player actions happen in encryption.

Why run this in FHE?

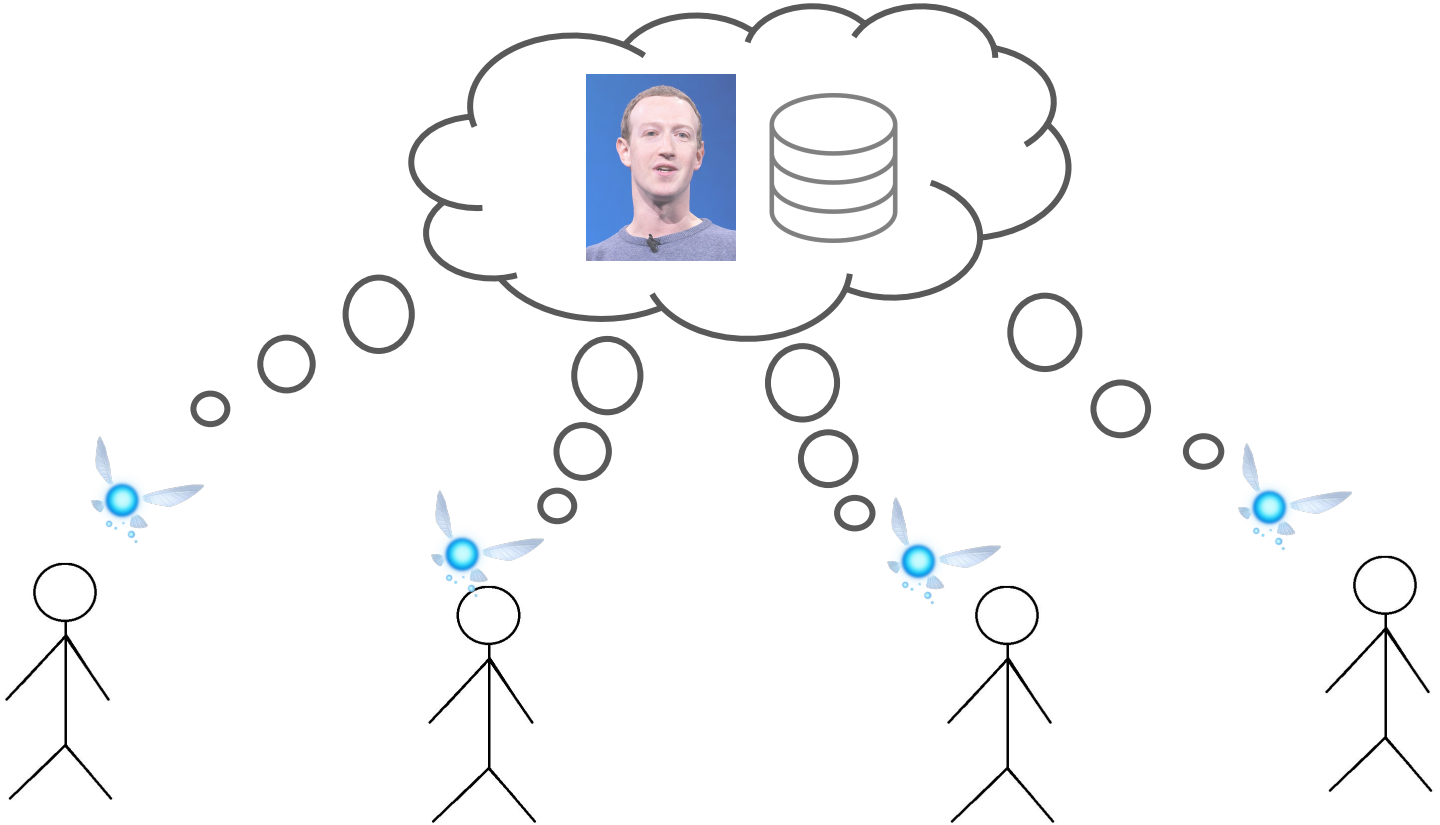
FHE enables the Frog Zone backend to run as a **hallucinated server**.

Hallucinated Servers

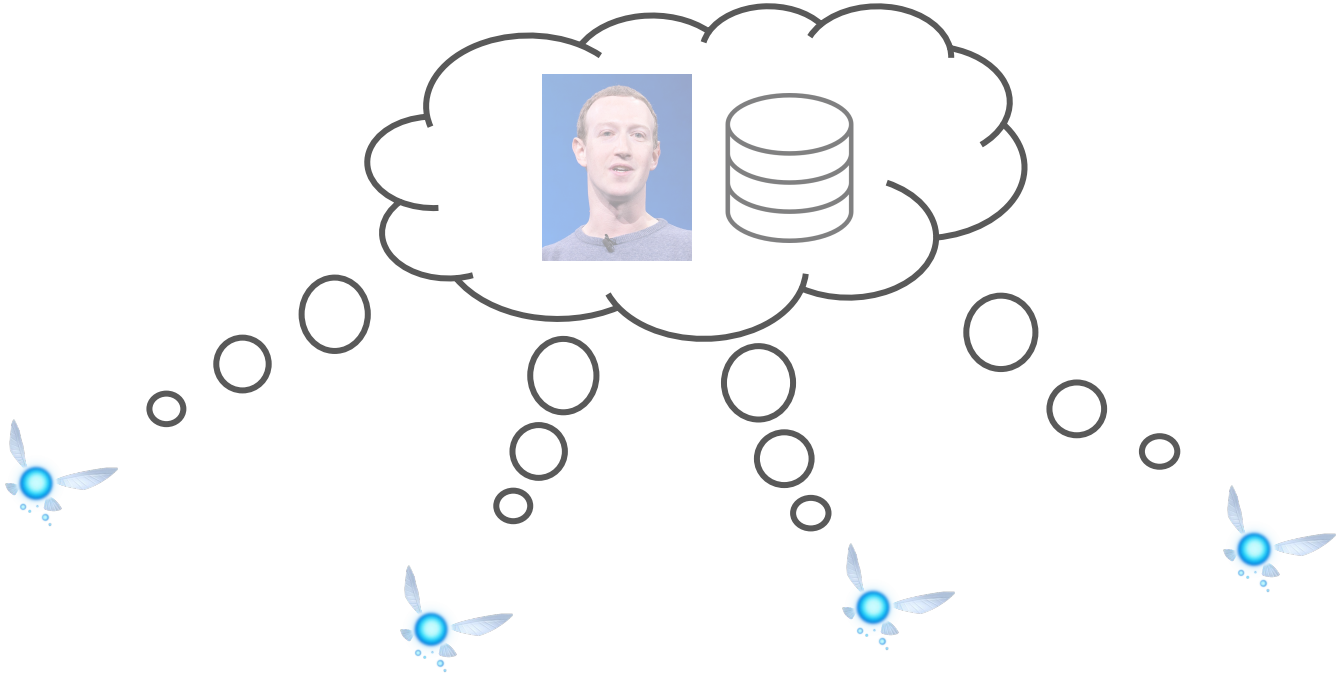
Applications today



What programmable cryptography can do



What programmable cryptography can do



What programmable cryptography can do

The long term promise

What if our digital services ran as hallucinated computations between just the relevant parties?

A server made of math that is perfectly secure, privacy-preserving, verifiable, interoperable, …

However, we’re very early in our journey…

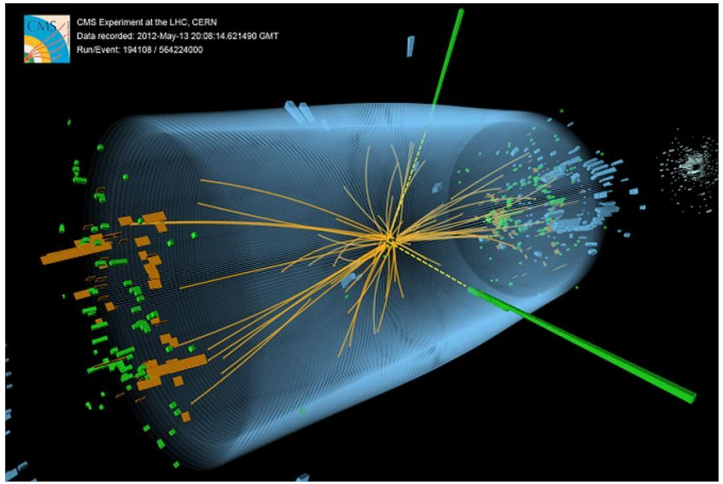
The game in numbers (part 2)

The game is running as a series of multi-party computations between 4 macbooks and 5 192-core AWS machines (c6a.48xlarge), costing about $200 / hr.

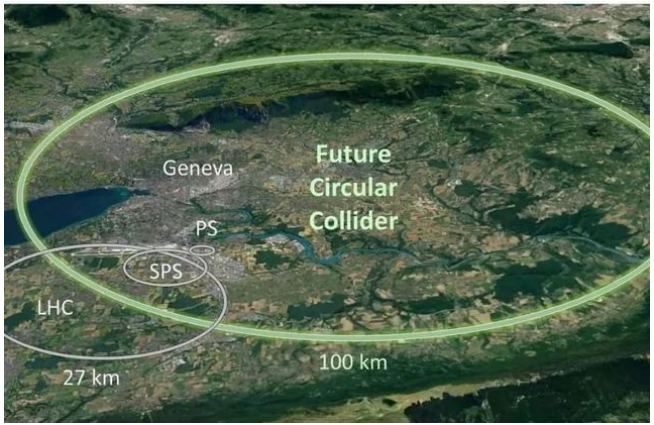
Every binary gate takes about 10ms to evaluate – this is 1,000,000,000 slower than regular computation.

For every bit of plaintext state, we store ~3,000 bits of ciphertext.

The game in numbers (part 2)



just one more collider bro. i promise bro just one more collider and we'll find all the particles bro. it's just a bigger collider bro. please just one more. one more collider and we'll figure out dark matter bro. bro cmon just give me 22 billion dollars and we'll solve physics i promise bro. bro bro please we just need to build one more collider t

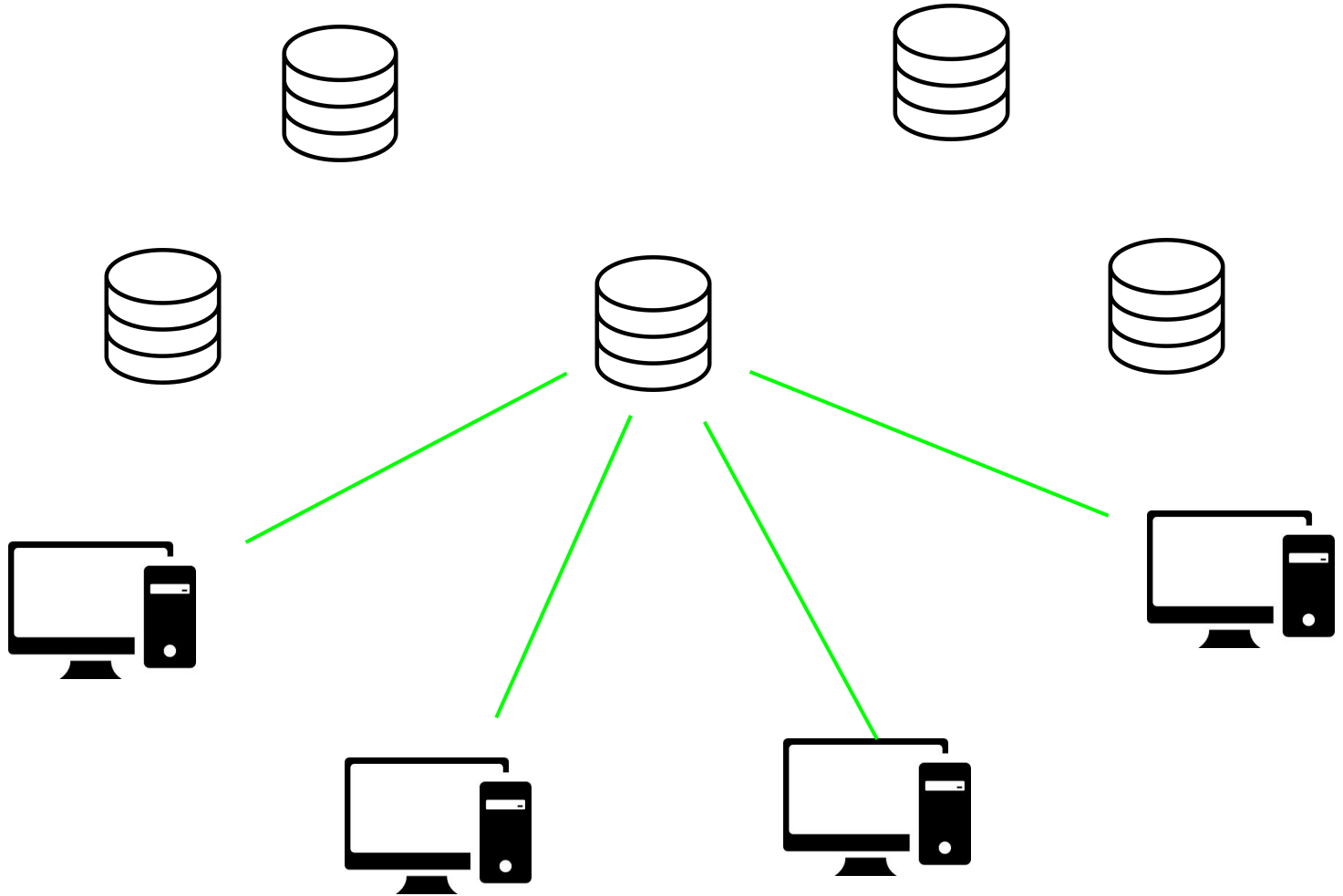


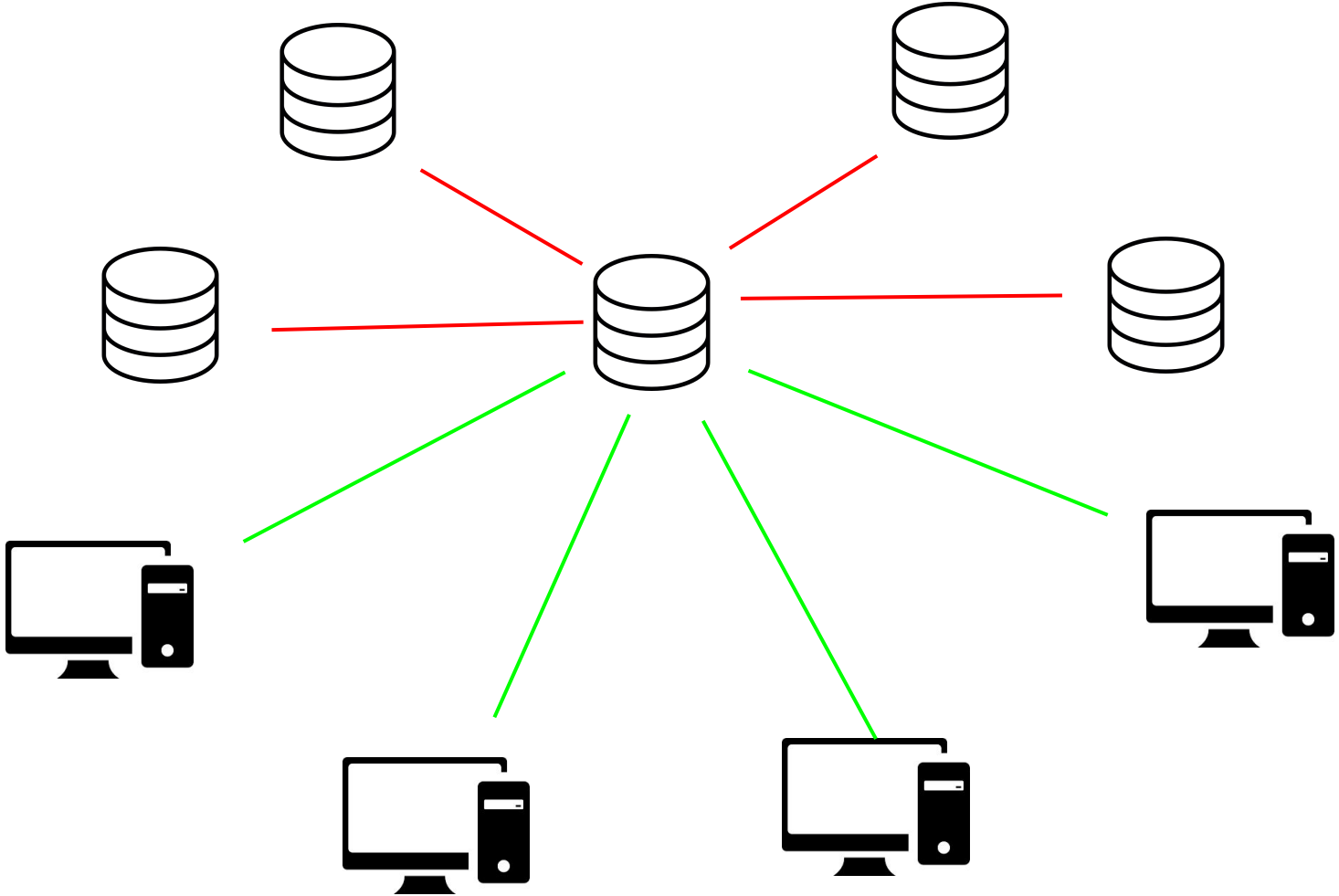
We have built the slowest and worst-performing game in the history of human computing.

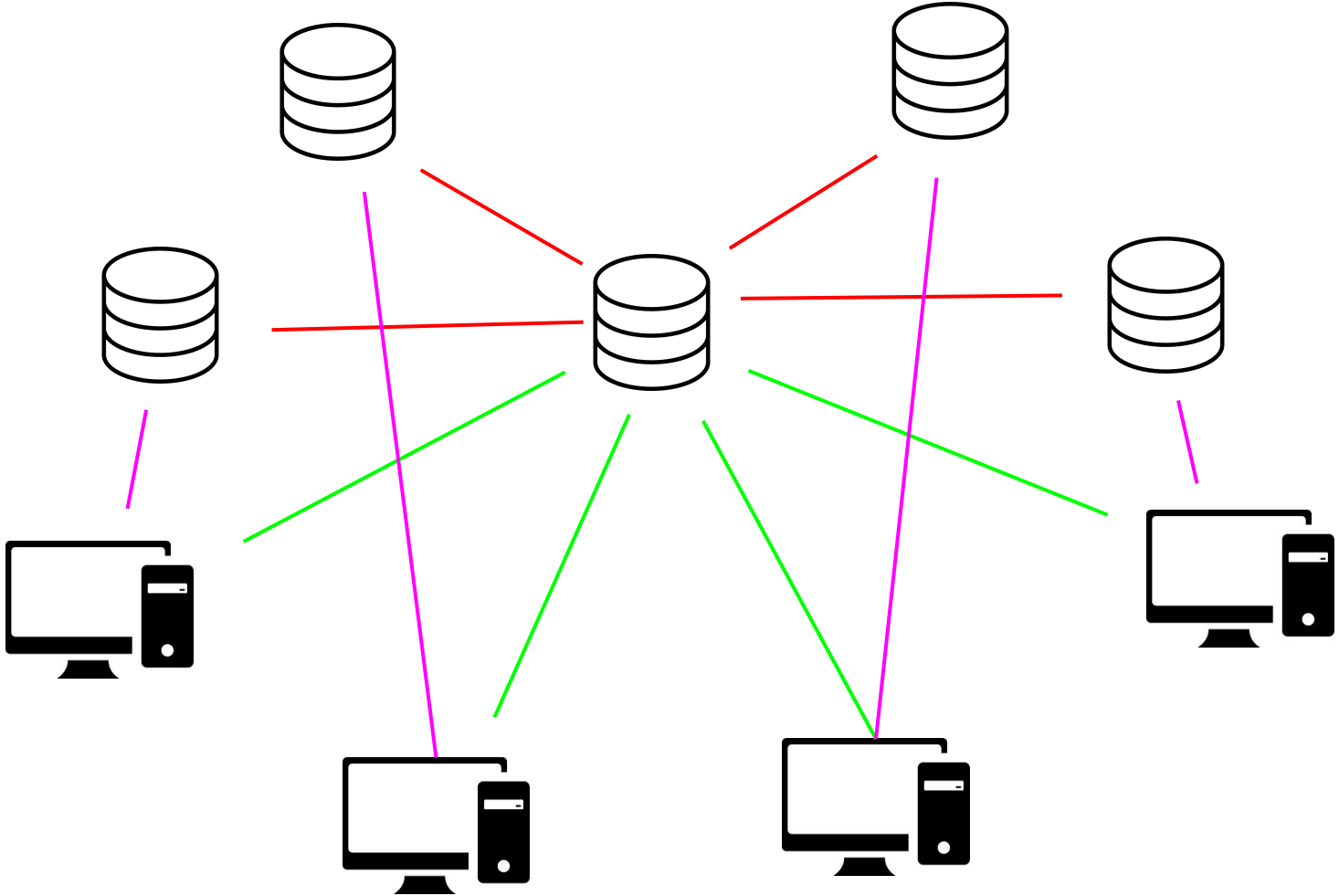
How does it work?

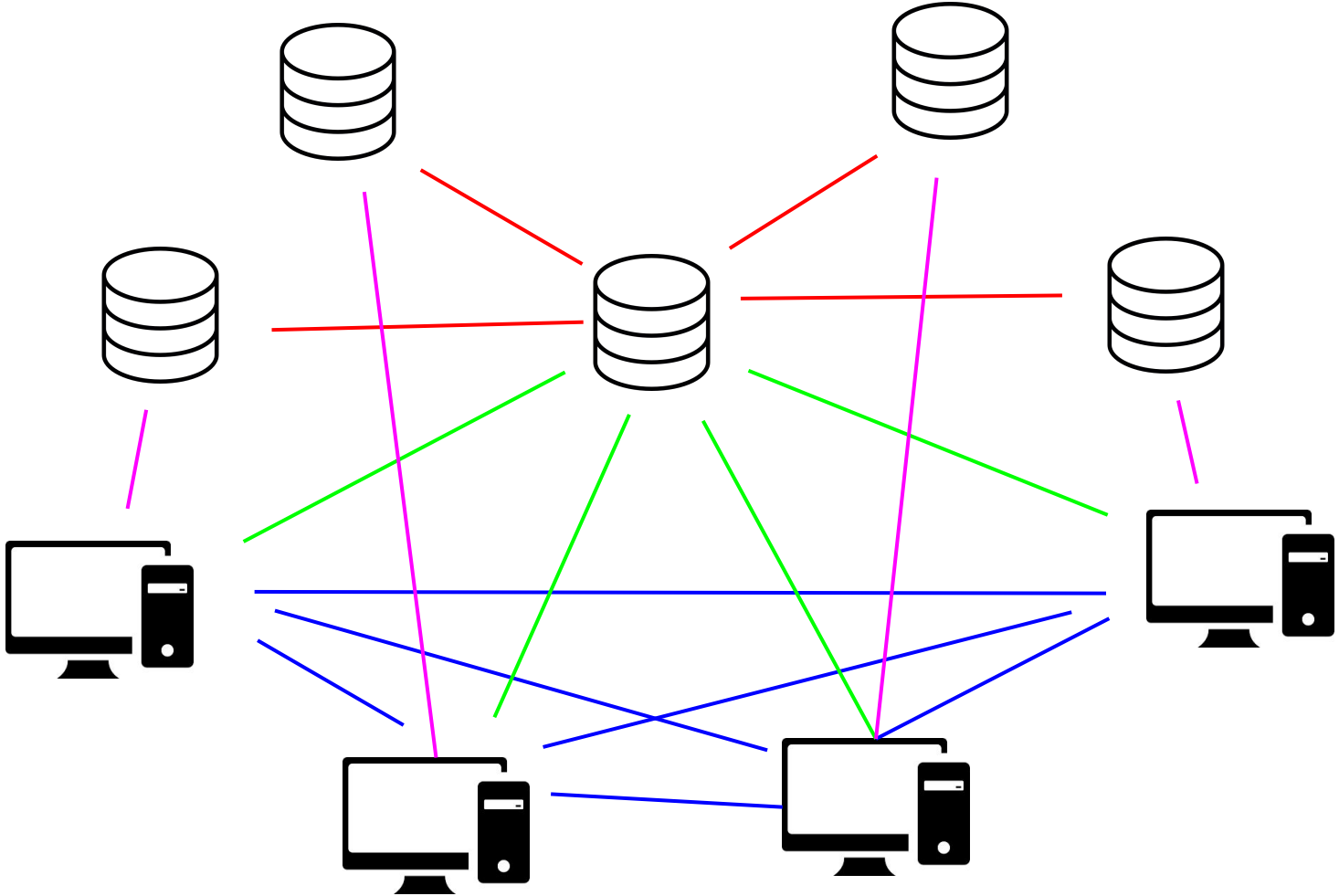
C6a.48xlarge
192 vCPU
384 GB memory

macbook







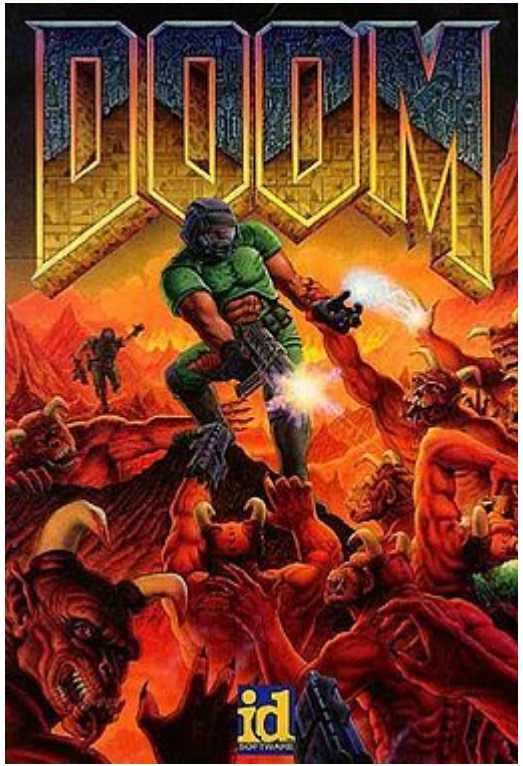


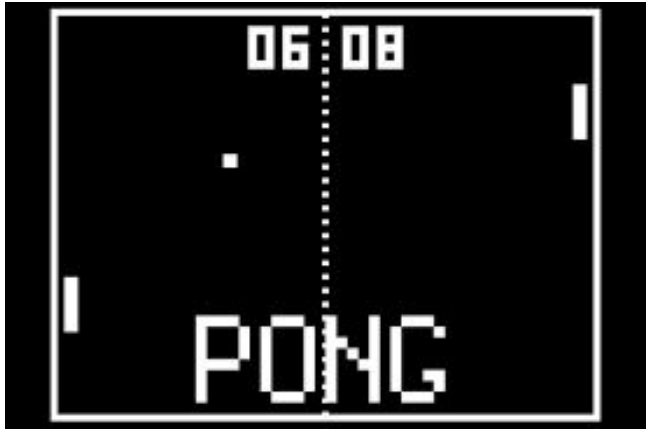
Are the servers trusted? Are they necessary?

Are the servers trusted? Are they necessary?

- No!
- They are performing only fully deterministic operations with no secrets.
  - They run on top of encrypted data using a deterministic FHE scheme.
- “Delegated computation”
  - Theoretically we could run all of these operations on the local machines, and have them cross-check each other.

Some Tricks


DOOM
id


06 08
PONG

Constraints

200 bytes of state

1,000,000,000 computation overhead

3,500 ciphertext overhead

Keep each player action operating under 1 second (so 4 seconds per “turn”)

Squeezing the most gameplay out of our constraints

Everything is a **move**–two encrypted bits that represent either Enc(UP), Enc(DOWN), Enc(LEFT), or Enc(RIGHT).

Tile	Item	Monster	Empty Water	Empty Land
What happens?	Update coordinate and apply item effects	Attack monster	Nothing	Update coordinate

Squeezing the most gameplay out of constraints

Every entity is ~32 bits, so no matter what is on a cell, we can send over 32 bits and deserialize differently

	Player	Item	Monster
Byte 1	Metadata (entity type, ID)	Metadata (entity type, ID)	Metadata (entity type, ID)
Byte 2	HP	Health upgrade	HP
Byte 3	ATK	Attack upgrade	ATK
Byte 4	Score	Points for picking up	Points for defeating

Moving monsters vs. moving players

Because we are working with circuits, every operation must be applied “simultaneously.” You can’t just check for a collision in your area, you have to check for collisions with **all** entities.

Three types of monsters:

- Immobile (like prickly bush)
- Ground monsters (like imps)
- Flying monsters (like bats)

Randomness (and what more we can get from it)

- At the beginning of the game and every turn, players share some encrypted randomness to the server.
- This randomness is mixed and used to determine how monsters move around and which monsters move.
- This means none of the players nor any delegated computation servers alone know what’s happening inside the Frog Zone.

What happens next?

What could we do with 10x more compute?

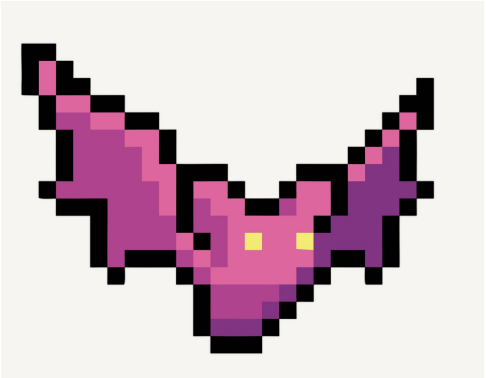
- Run things locally
- Procedurally generate the map itself

Capabilities, not compute

- Memory
- Obfuscation and Functional Encryption
- Verifiability
- Put it on-chain!

Other questions

- Another “Autonomous World”?
- What is it like to be a bat?
  - Mind upload into the Frog Zone
  - I’m gonna get encrypted

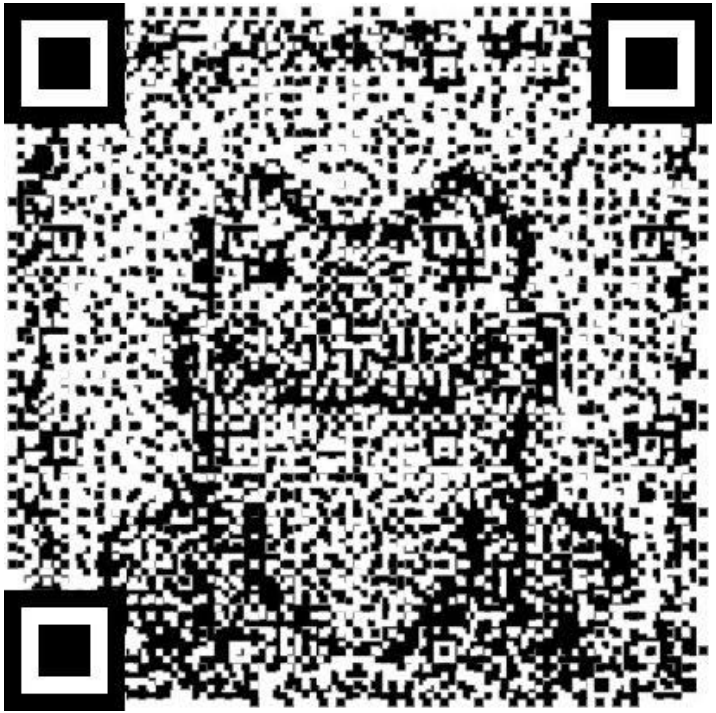


Acknowledgements

Contributors

- Janmajaya (Gauss)
- Han, Edu, CC (PSE)
- Sohan (Small Brain Games)
- Marianne (Lazer)
- Justin, Michael, Arturo (0xPARC)

Thank You!



a froggy gift