

# Accountable and privacy preserving data processing via distributed ledgers

National and Kapodistrian University of Athens  
Theoretical Computer Science

Christos Nasikas

# Outline

- Introduction
- Architecture
- Future work
- Related work



**I need patients data for medical research!**



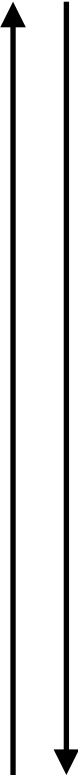
**Sure, here you are!**

**GDPR makes  
that illegal**

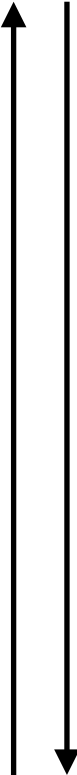
How can we help Dr. House  
help the patients ?



Blockchain



Data Controller

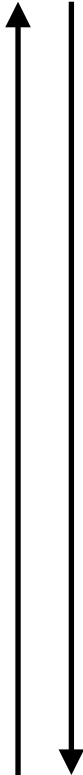


Data Requestor





Blockchain



Data Controller



Data Processor

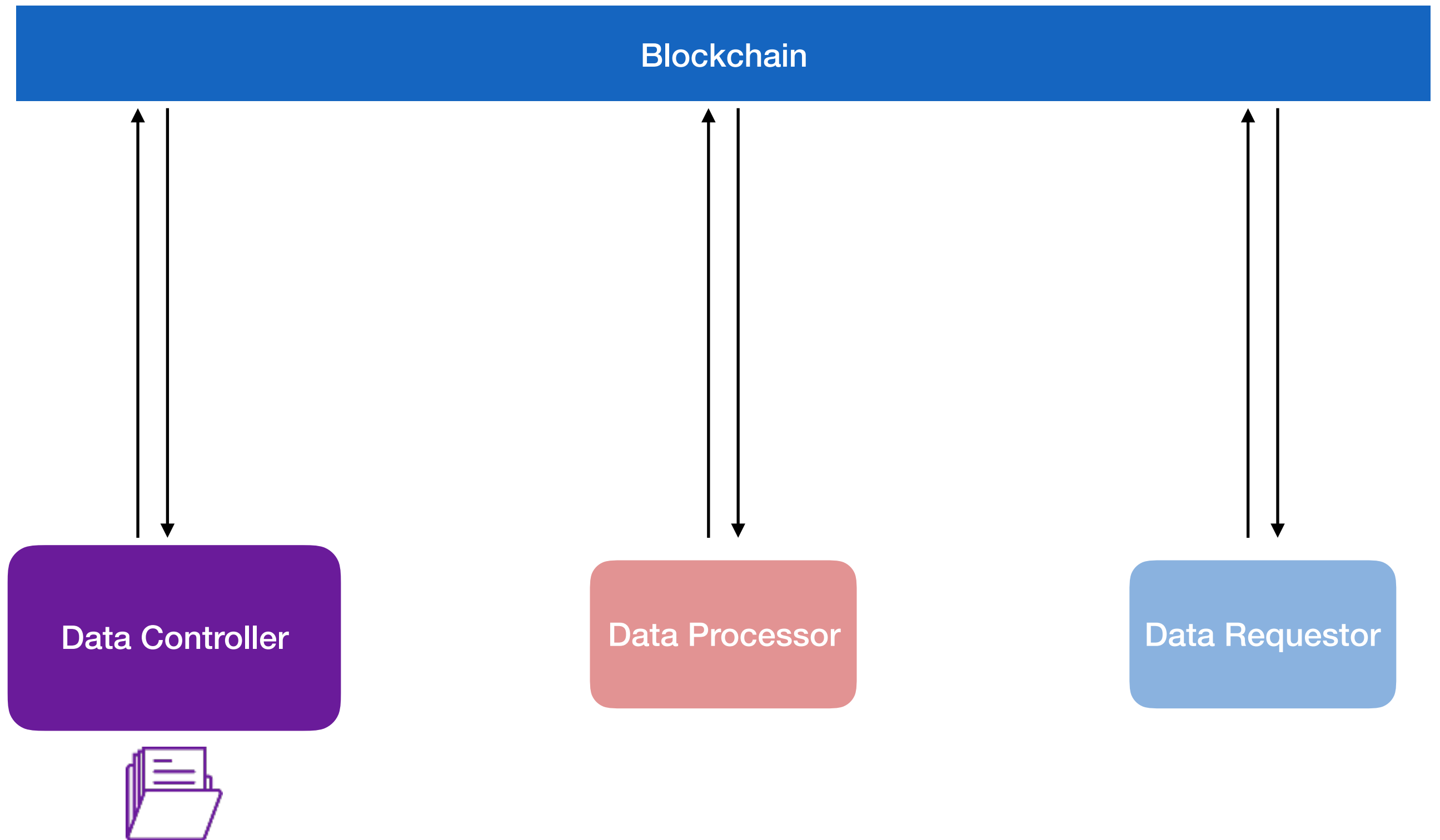


Data Requestor



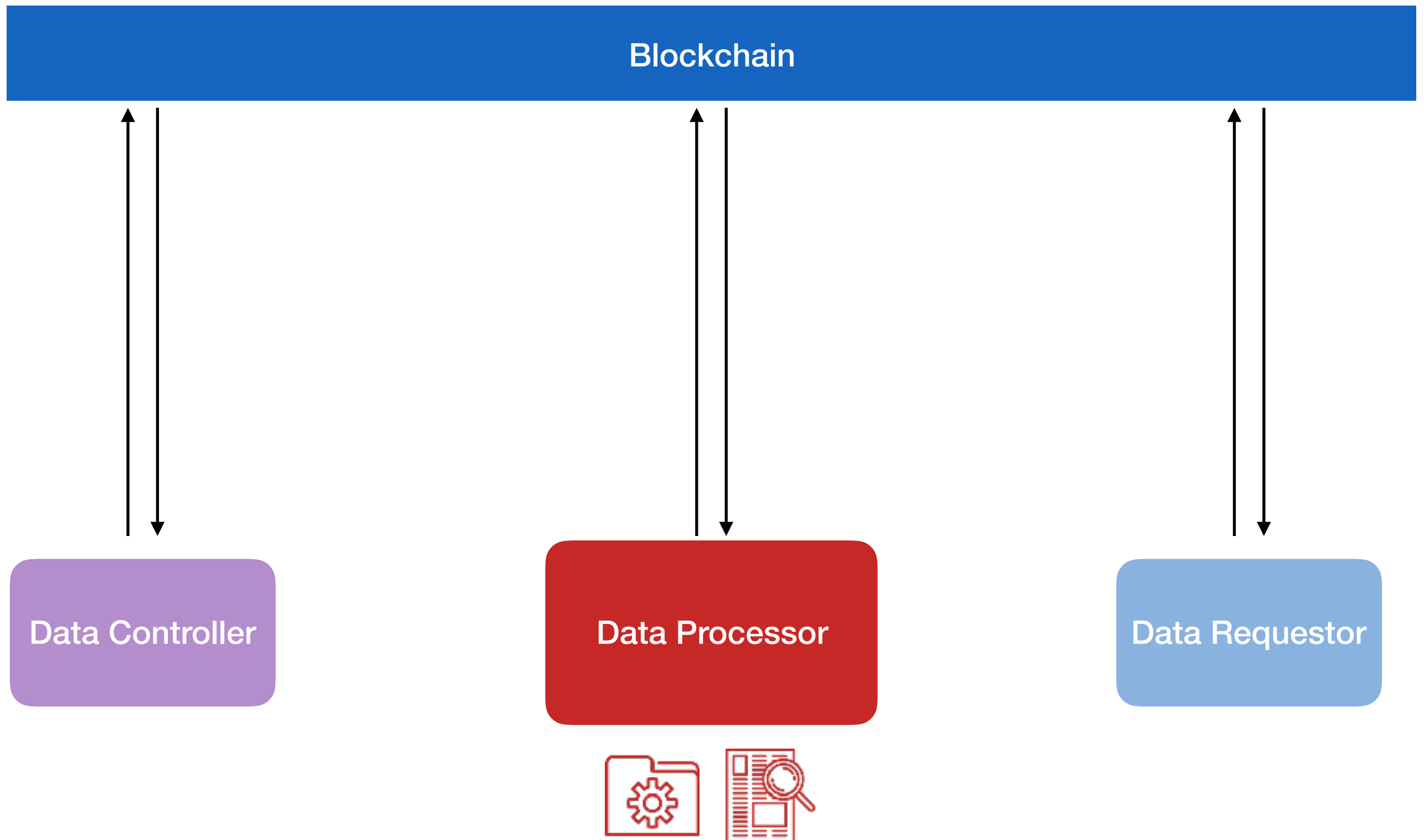
Architecture

# Participants

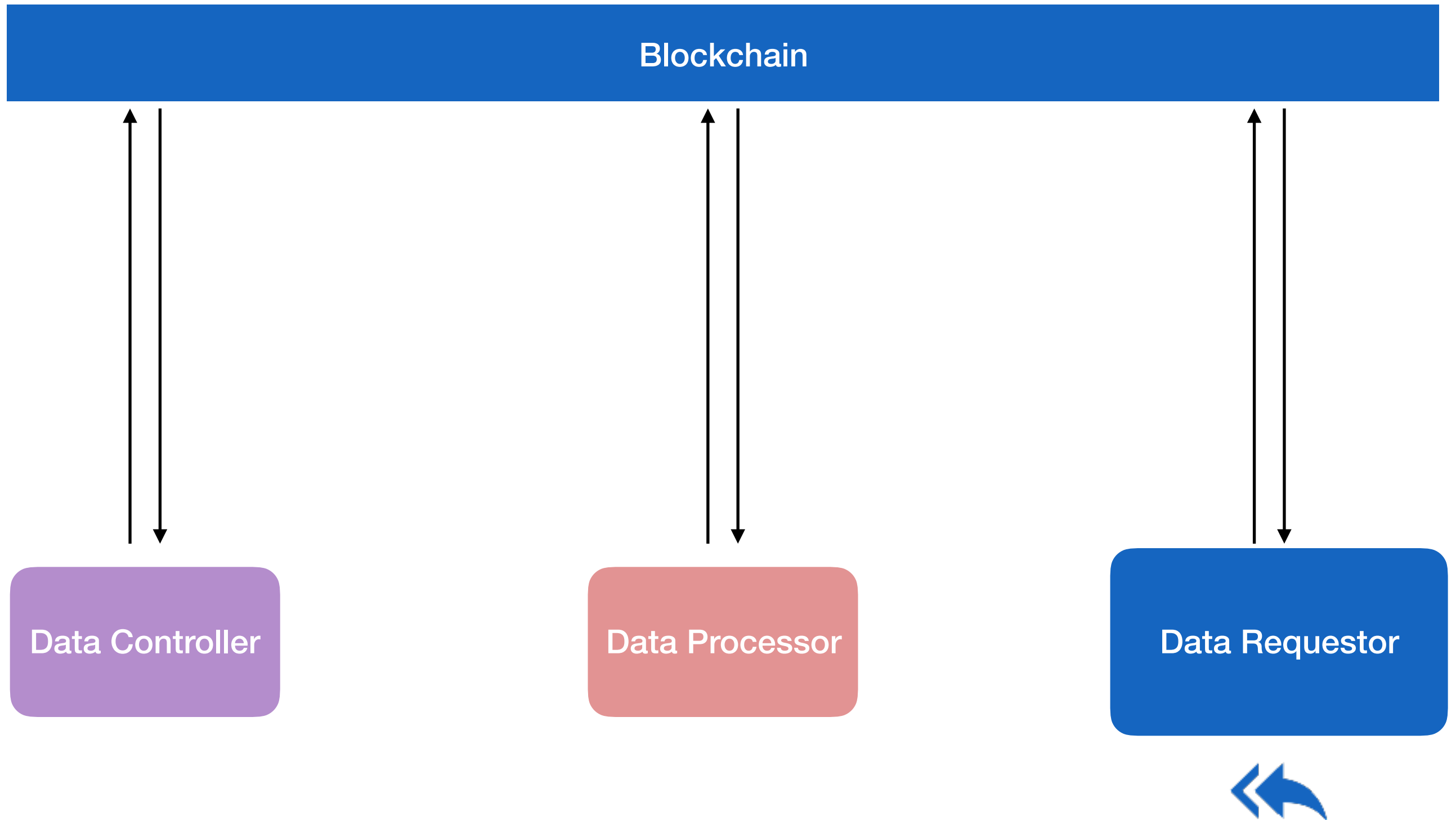




# Participants



# Participants



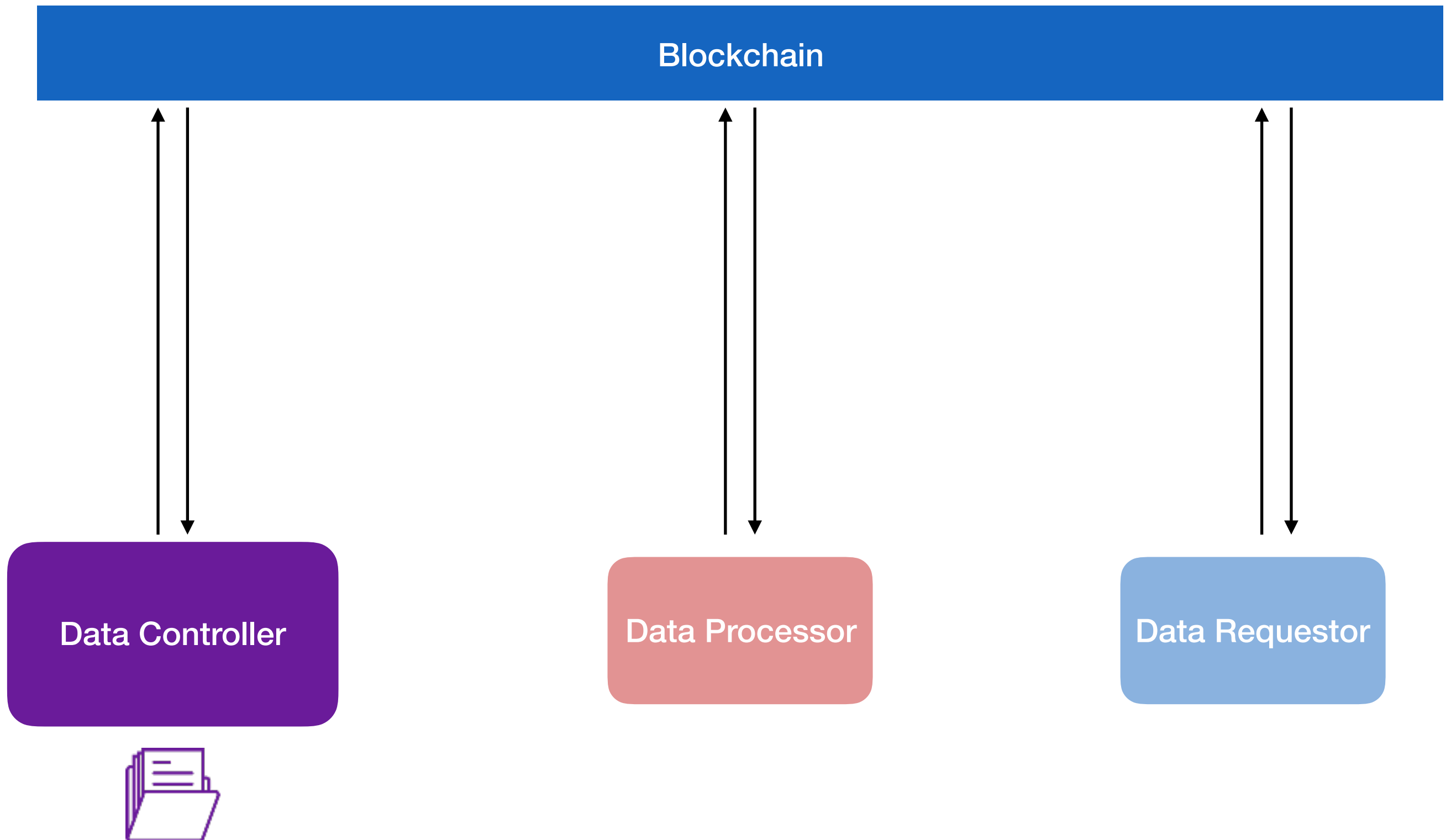
# Blockchain advantages

- Immutability
- Accountability
- Auditability
- Non-repudiation
- Public bulletin board
- Secure decentralized time-stamping

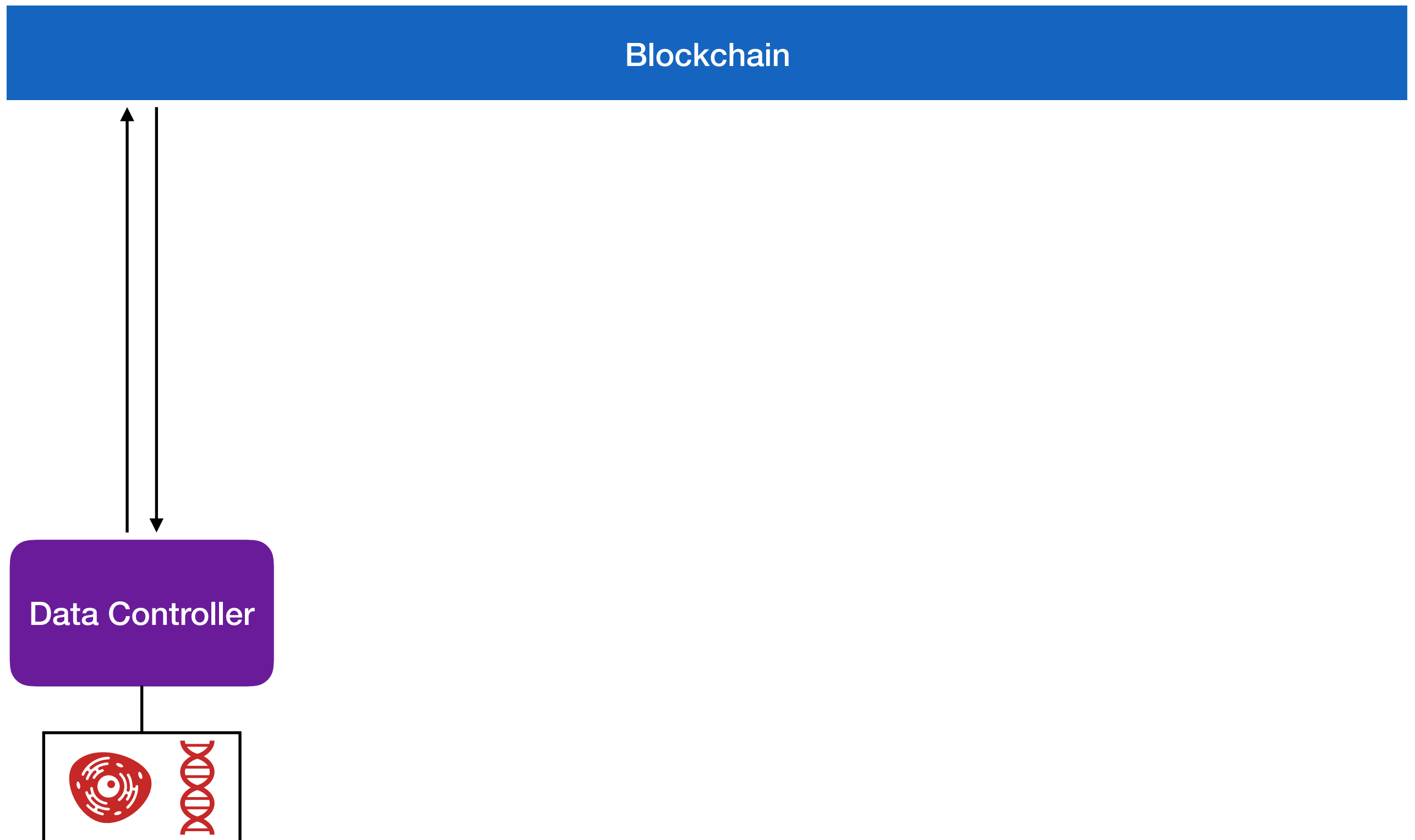
# Algorithms

- Sum
- Average
- Count
- Maximum / Minimum
- Median

# Data registration

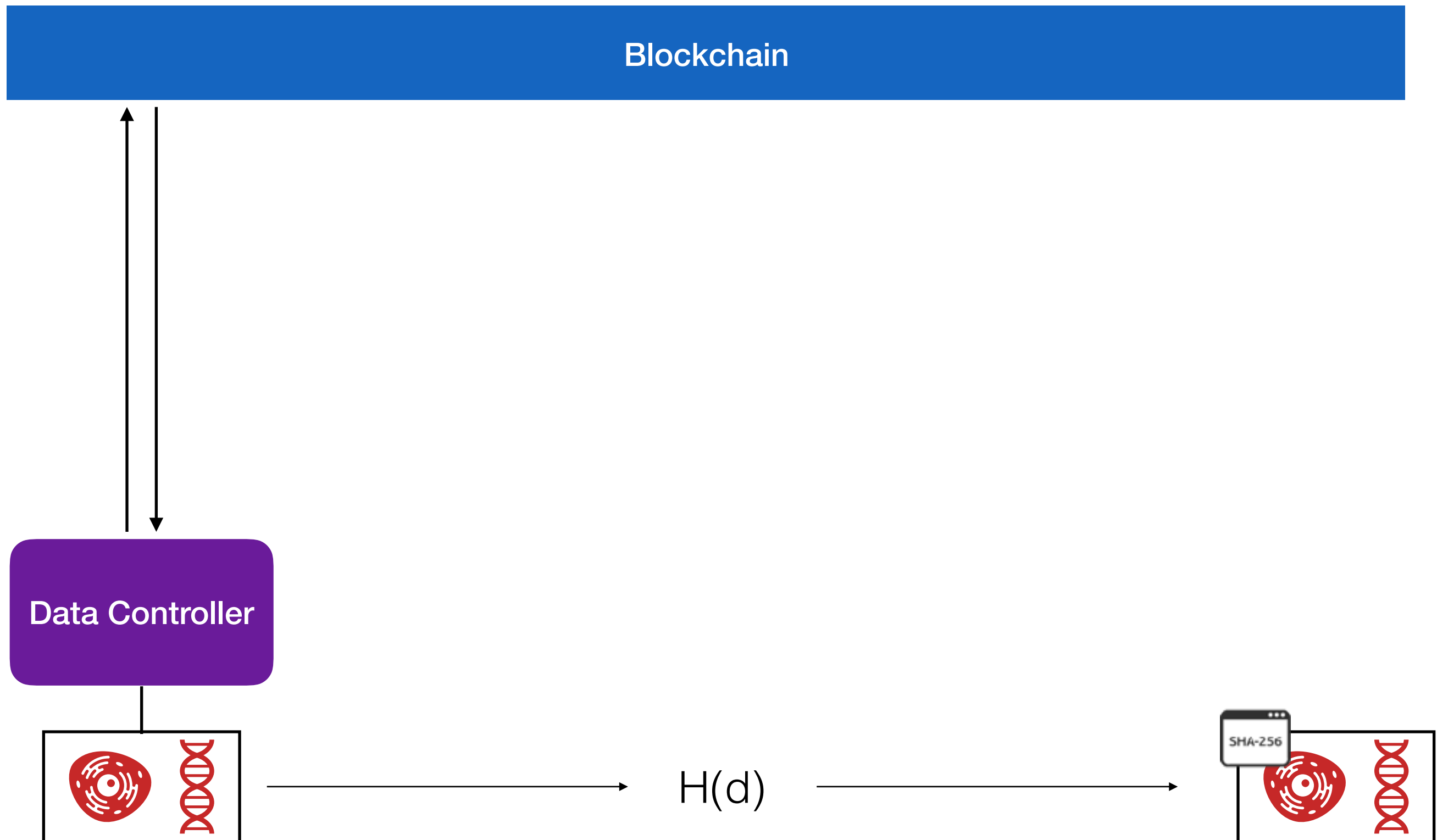


# Dataset Registration



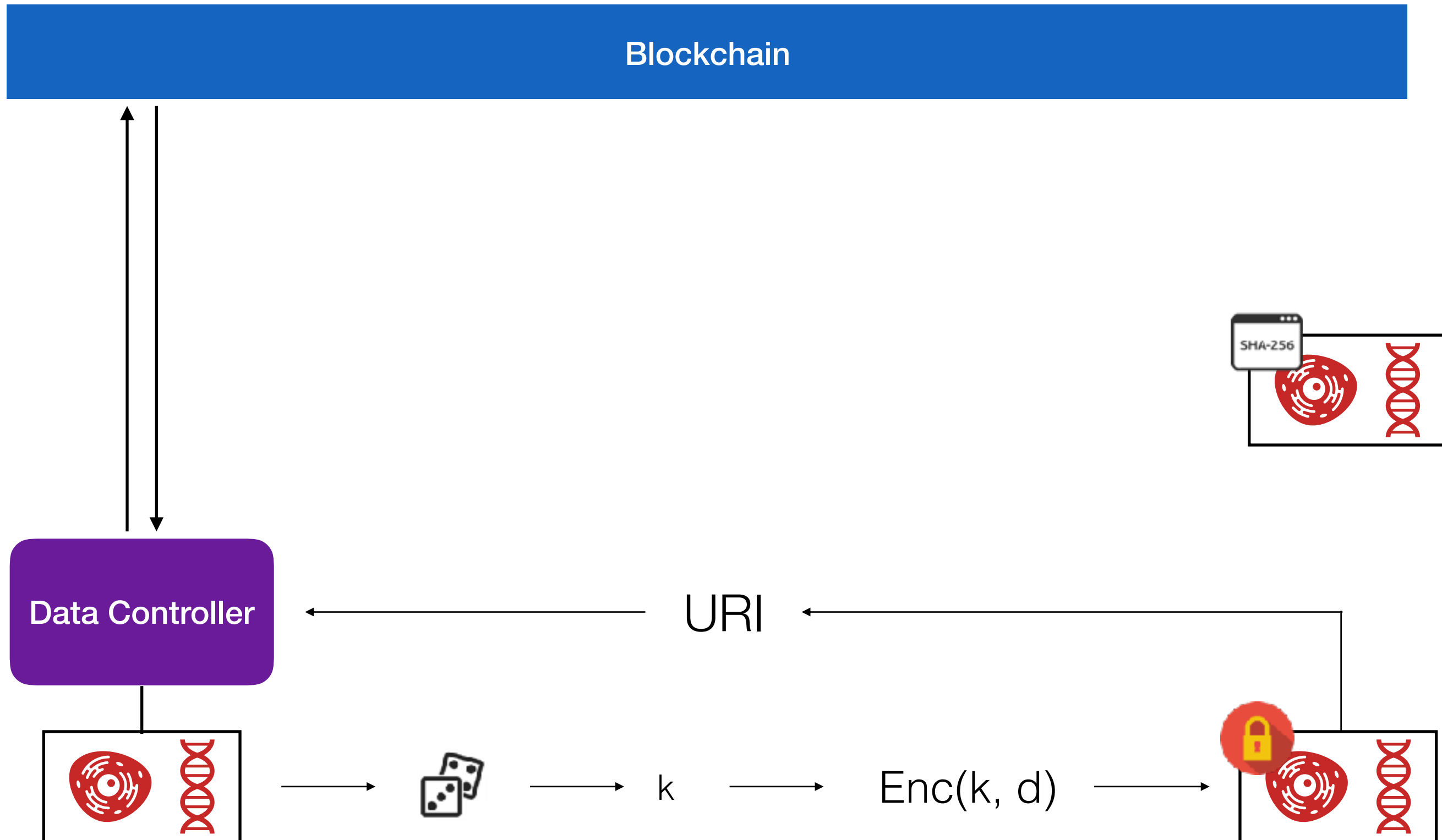
# Dataset Registration

Step 1:  
Hash dataset



# Dataset Registration

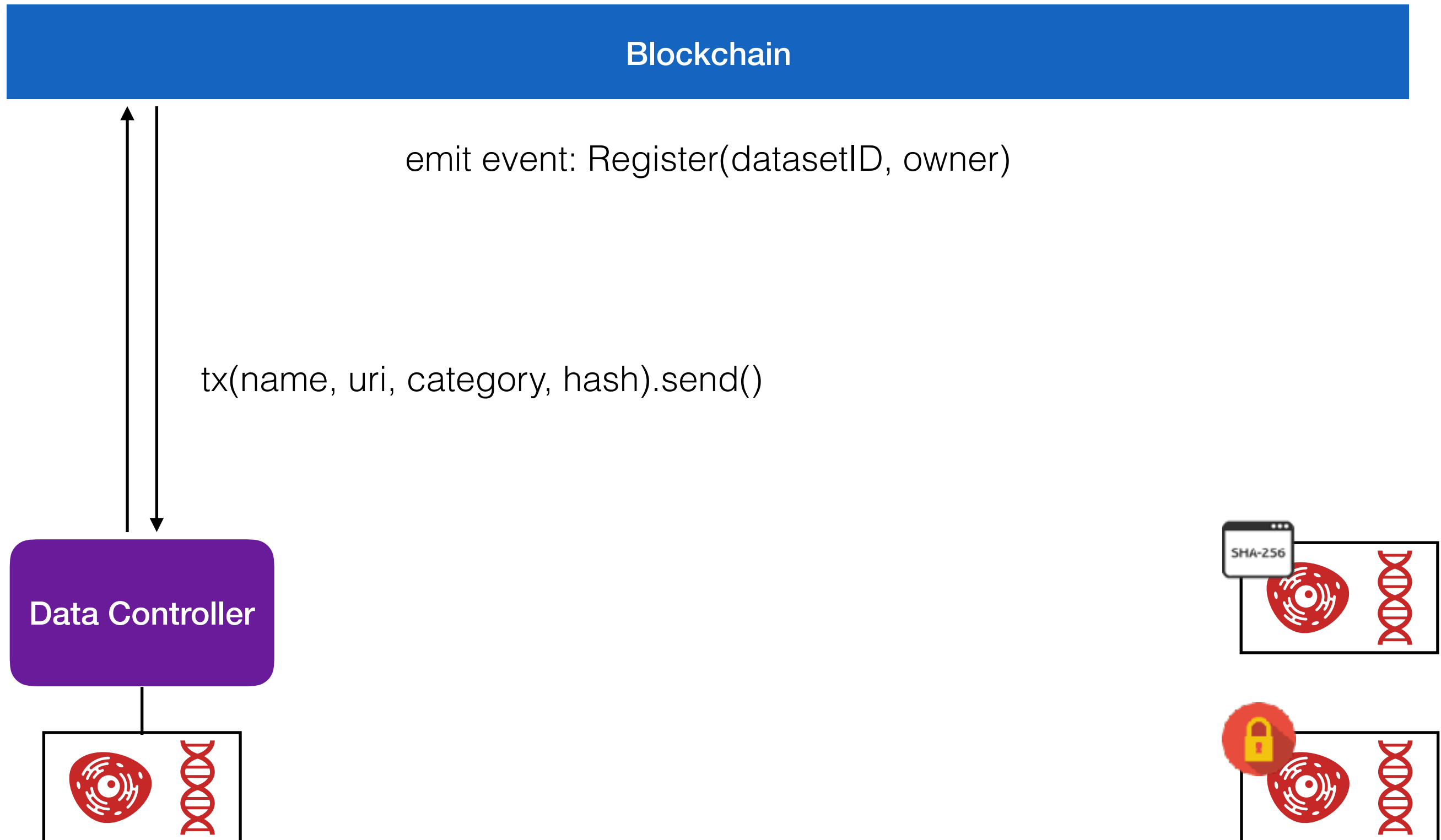
Step 2:  
Encrypt dataset



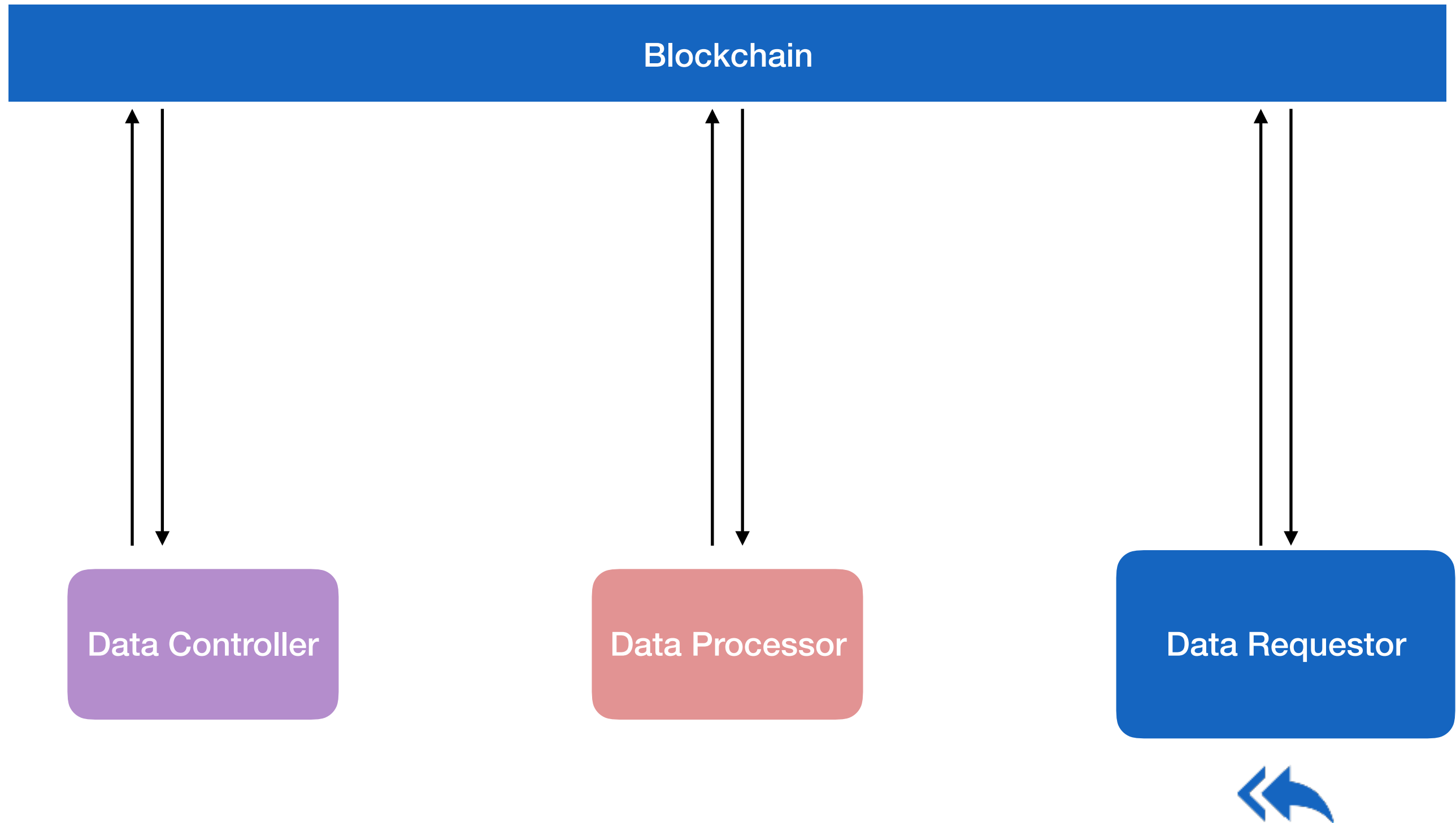


# Dataset Registration

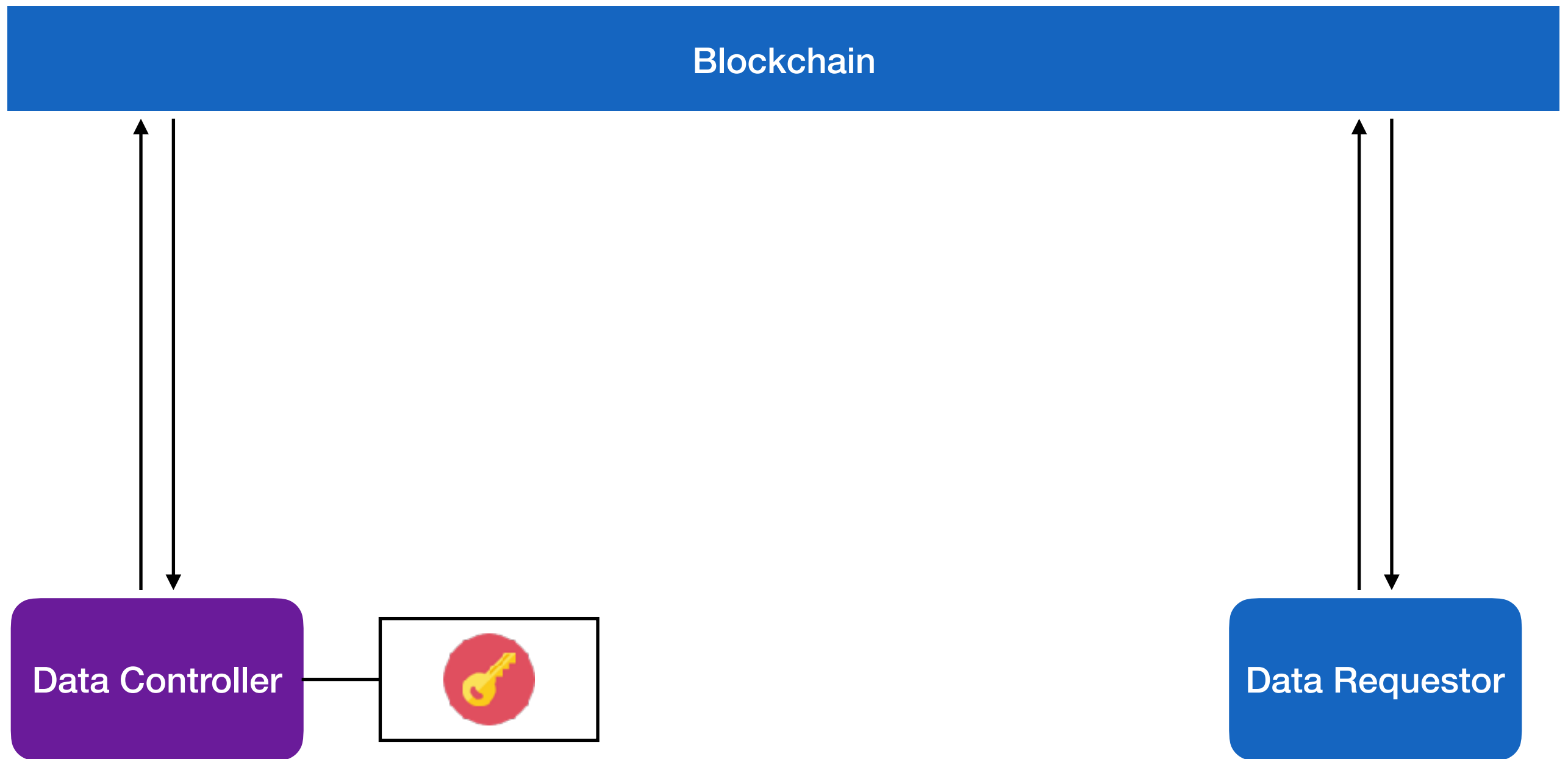
Step 3:  
Register dataset



# Request for processing



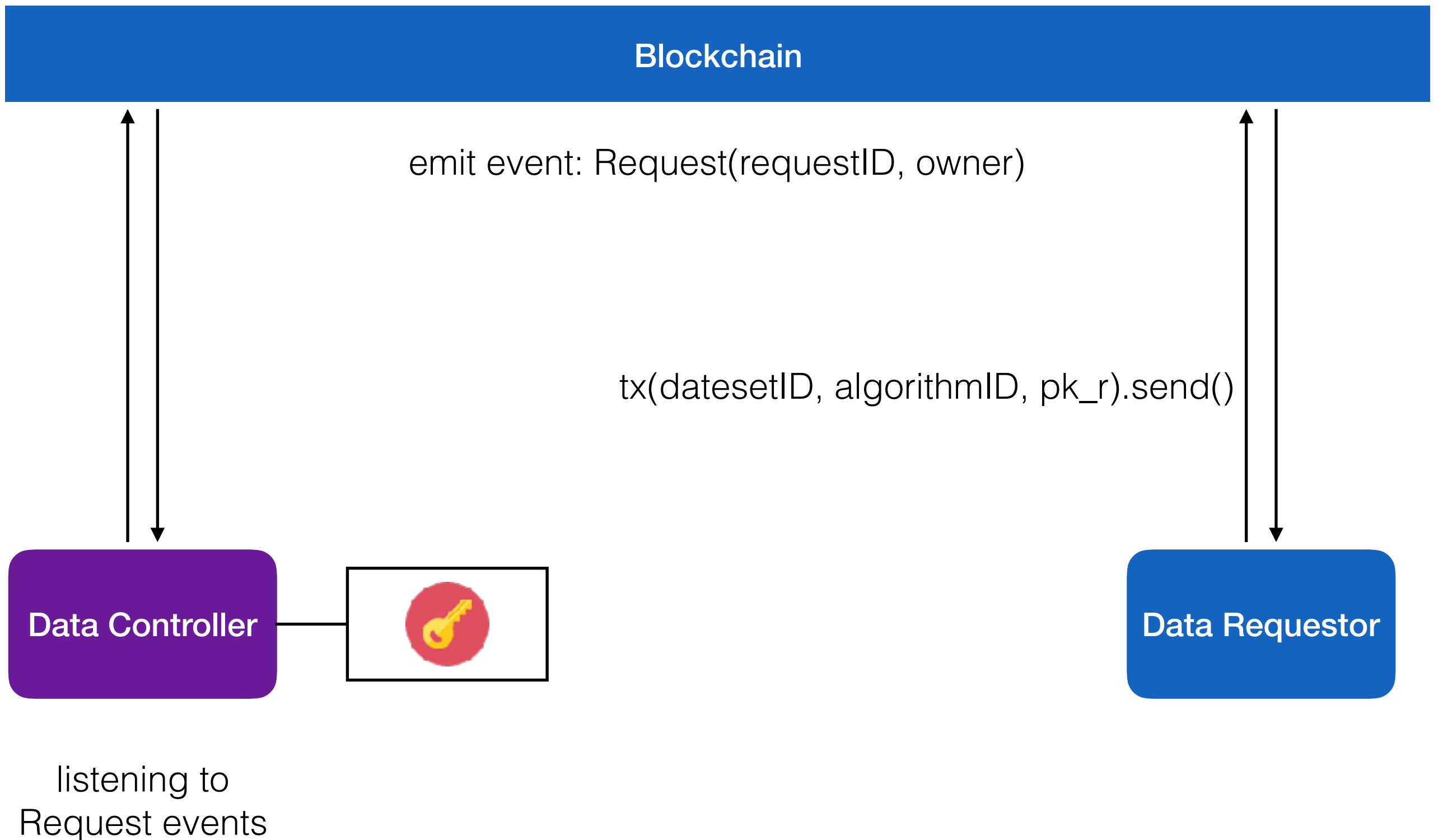
# Request for processing



listening to  
Request events

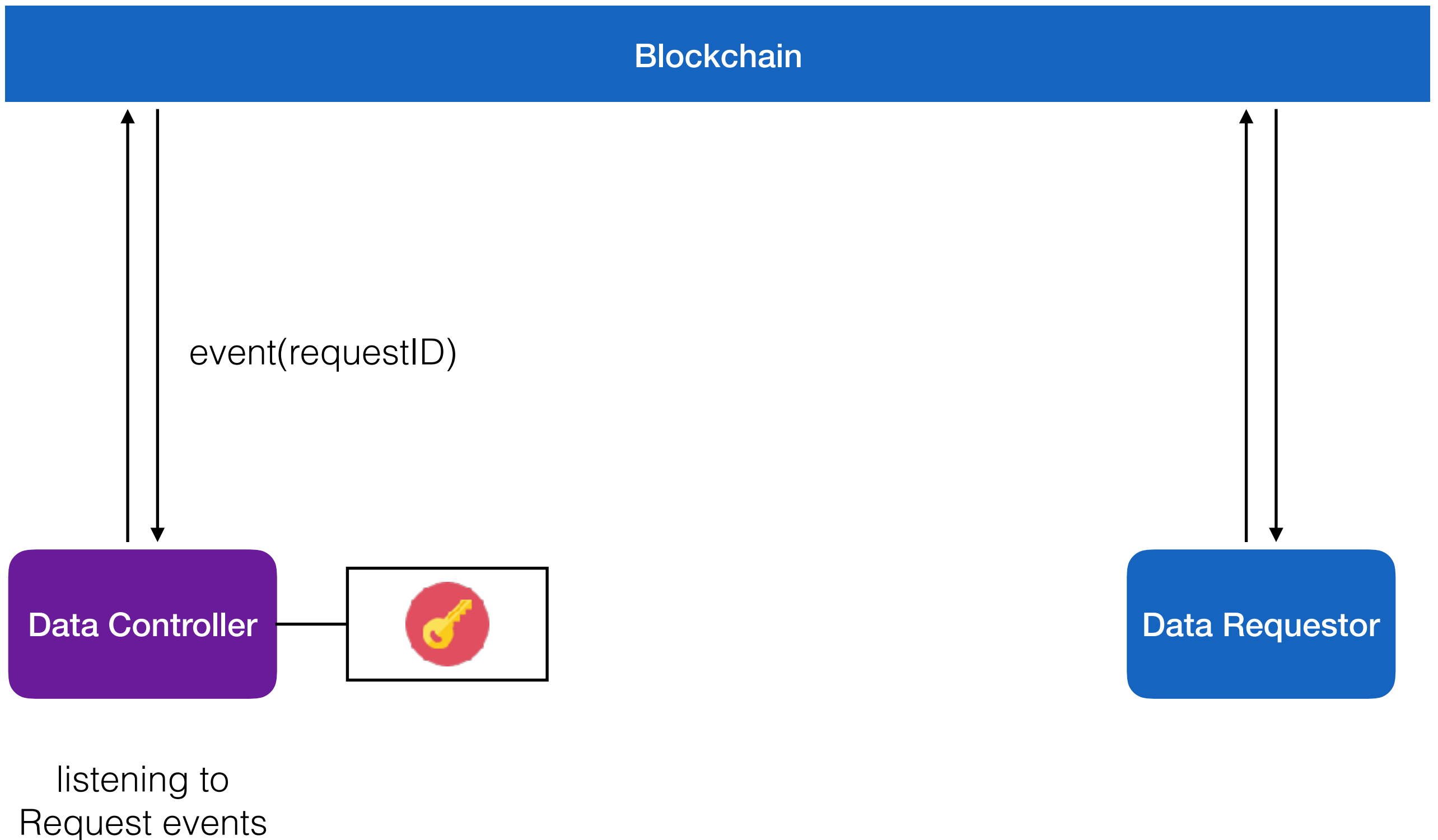
# Request for processing

Step 1:  
Request for dataset processing



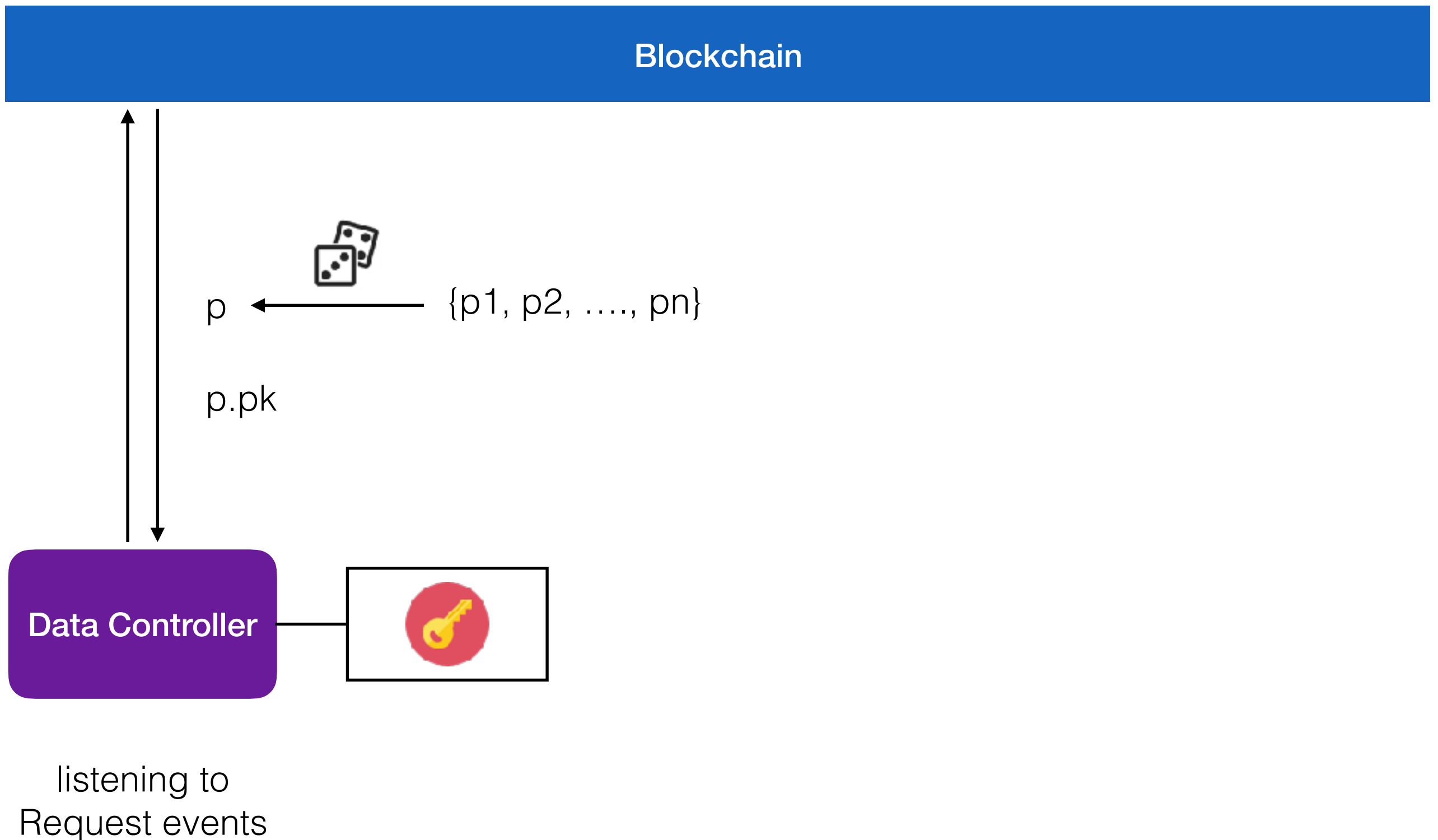
# Request for processing

Step 1:  
Request for dataset processing



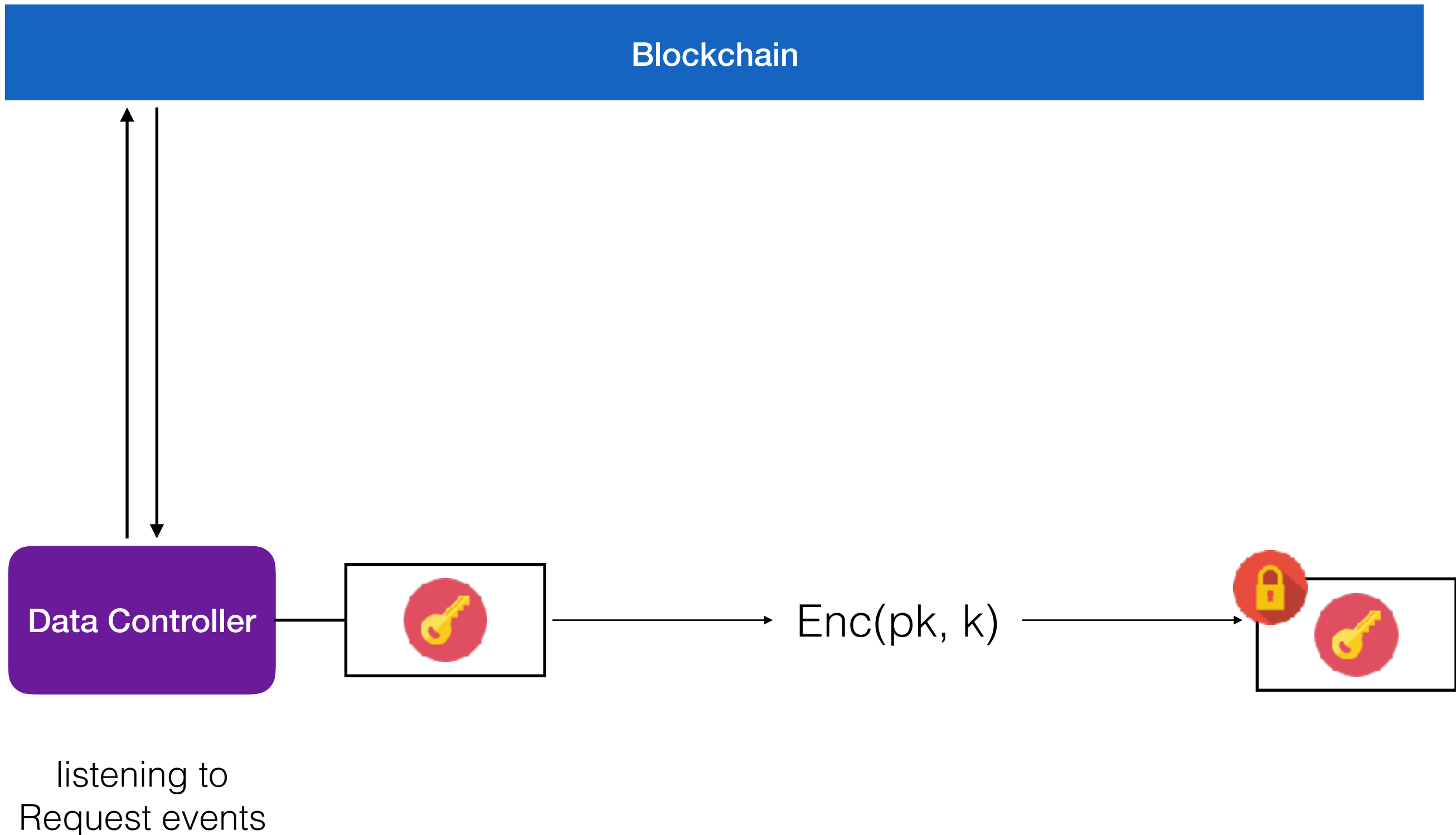
# Request for processing

Step 2:  
Get processor's public key



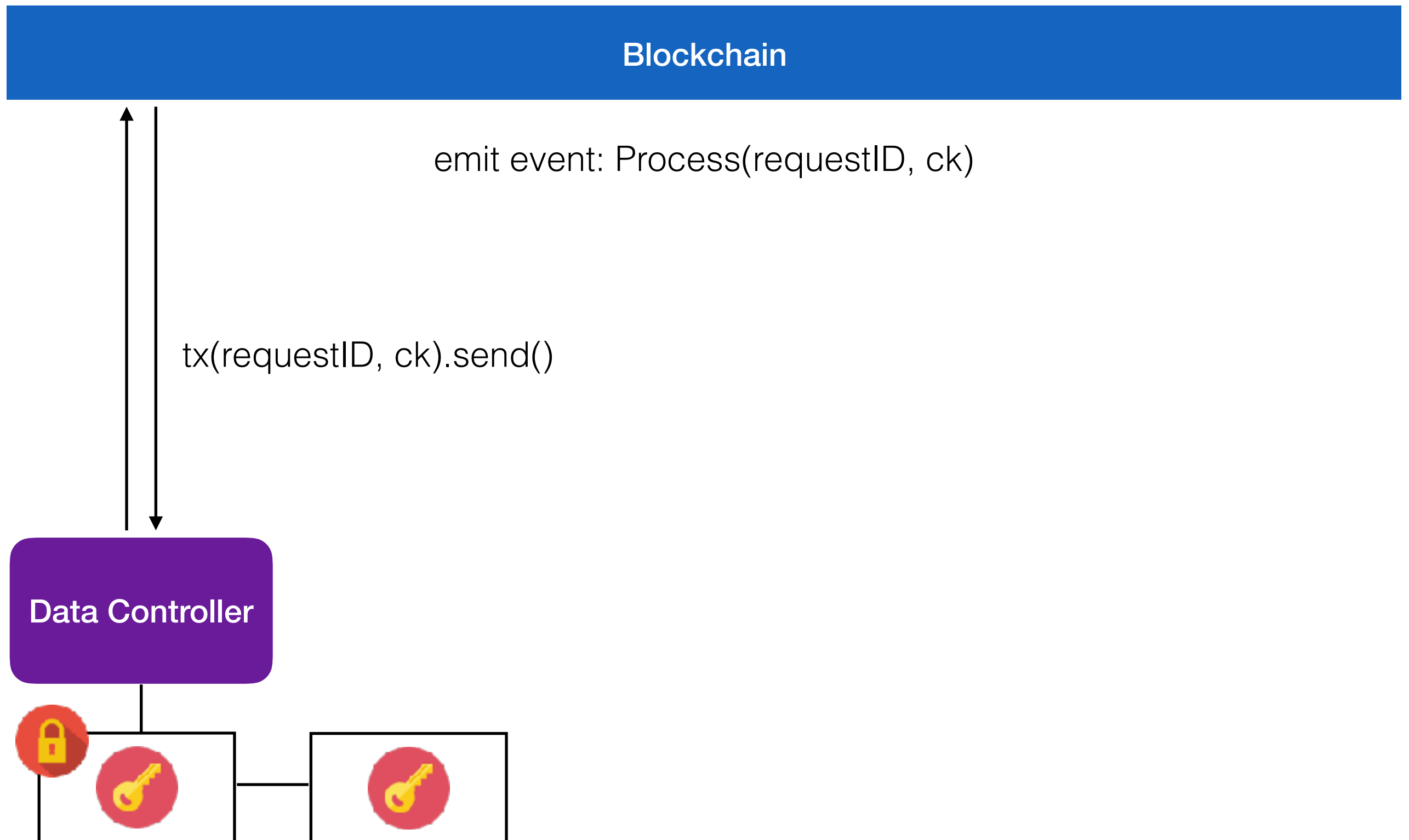
# Request for processing

Step 3:  
Encrypt symmetric key



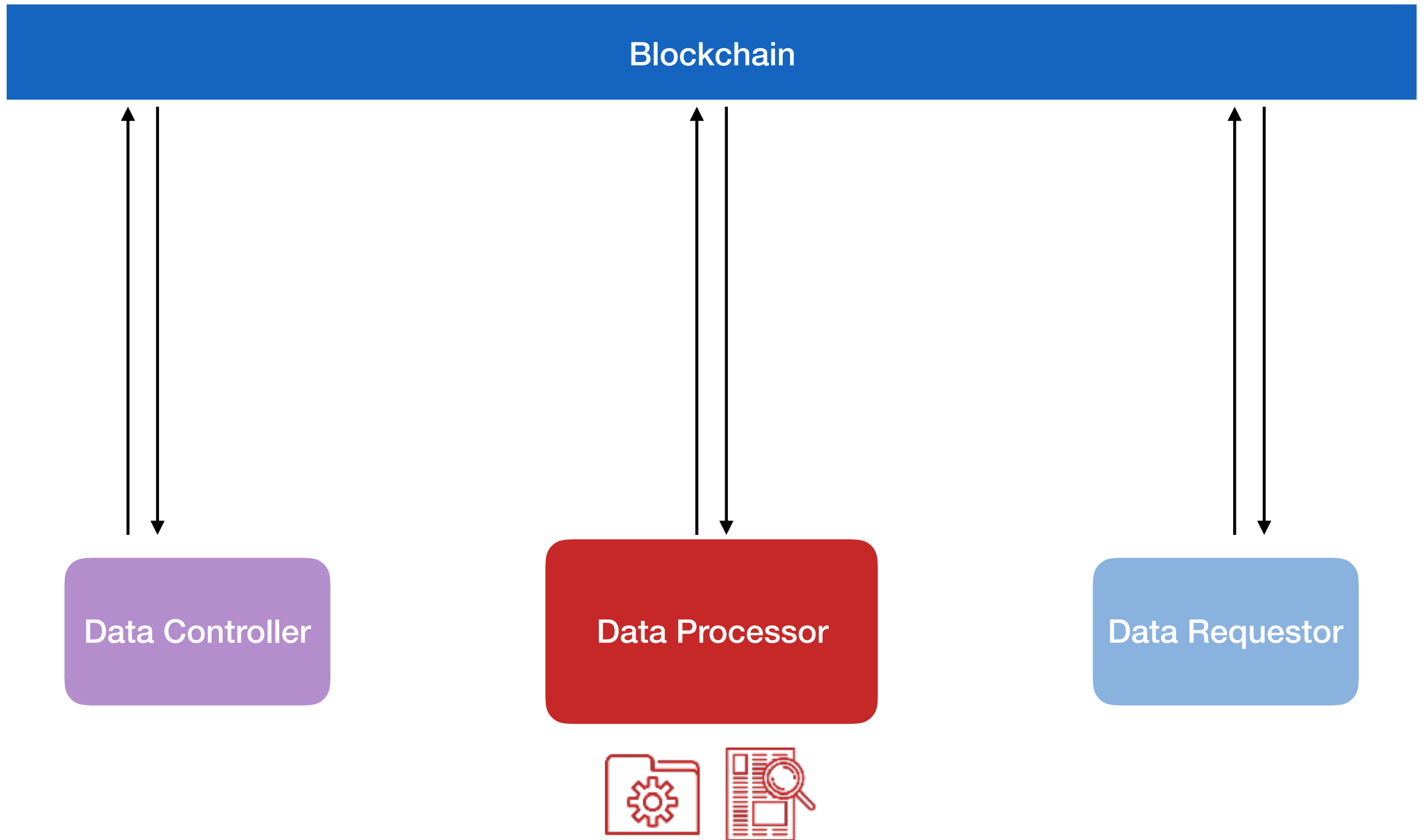
# Request for processing

Step 4:  
Send symmetric key to processor





# Processing



# Processing

Blockchain

```
graph TD; Blockchain[Blockchain] <--> DP[Data Processor]; DP --- L[listening to Process events];
```

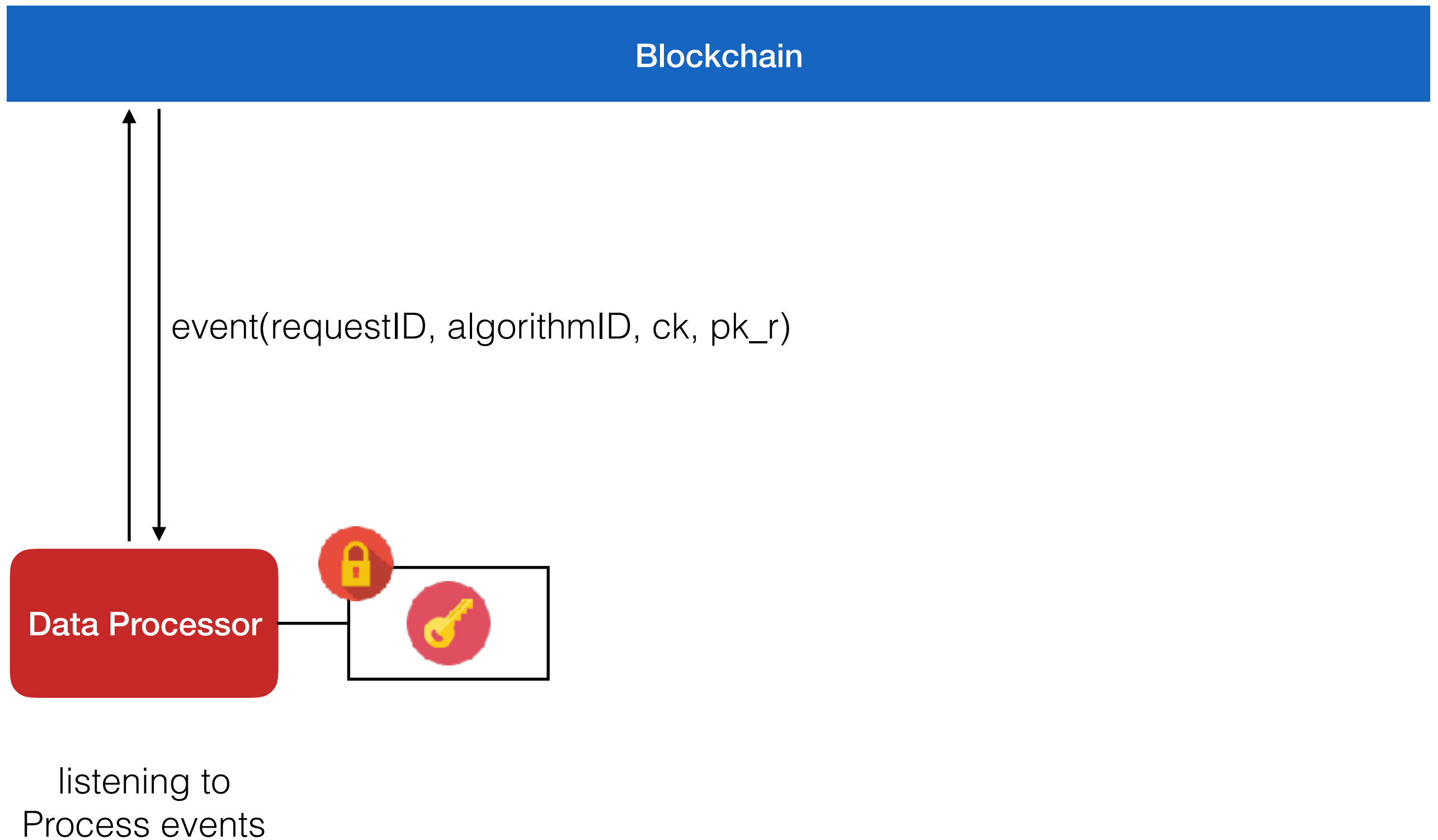
The diagram illustrates the interaction between a Blockchain and a Data Processor. At the top, a blue horizontal bar represents the Blockchain. Below it, a red rounded rectangle represents the Data Processor. A vertical double-headed arrow connects the Blockchain to the Data Processor, indicating bidirectional communication. Below the Data Processor, the text 'listening to Process events' is written.

Data Processor

listening to  
Process events

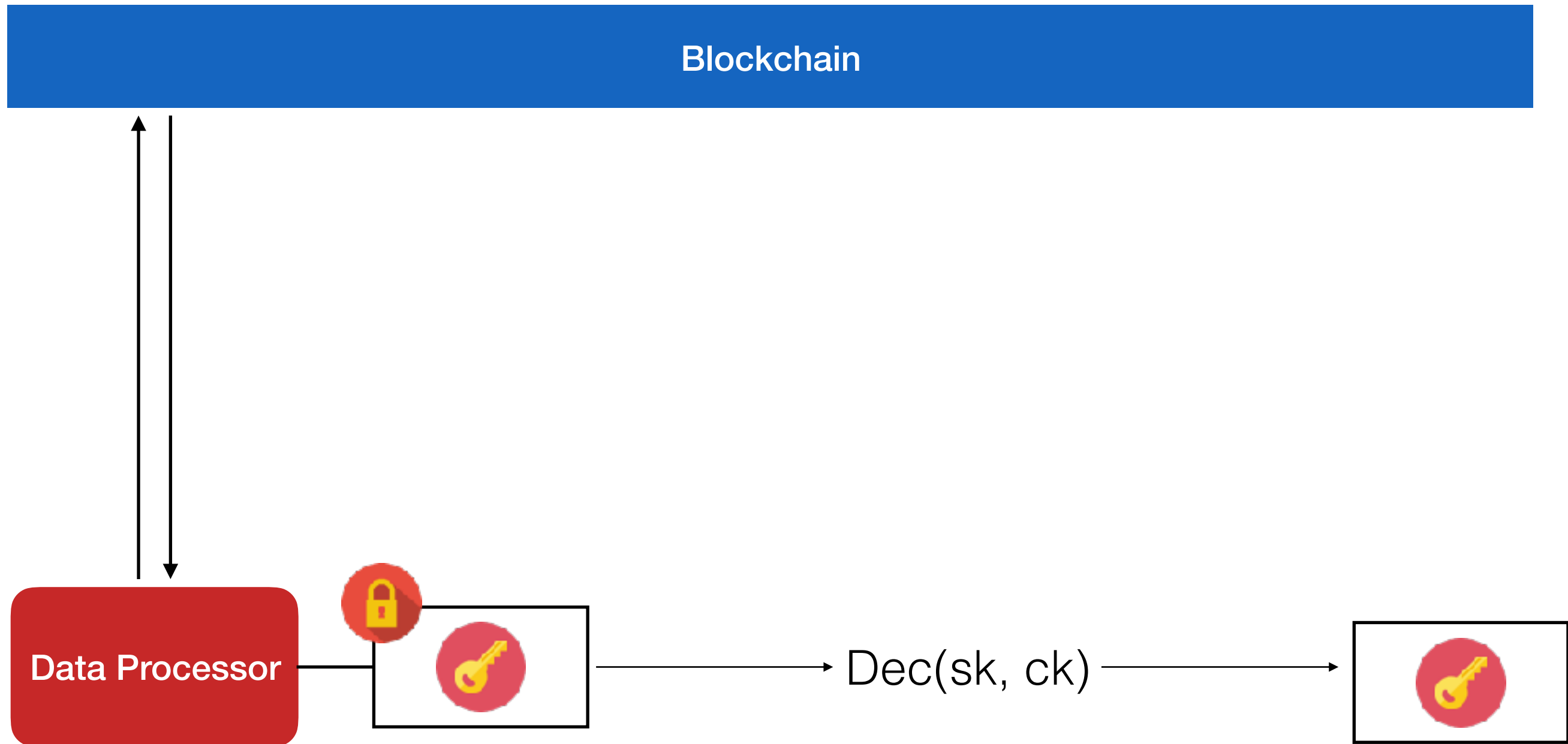
# Processing

Step 1:  
Catch process event



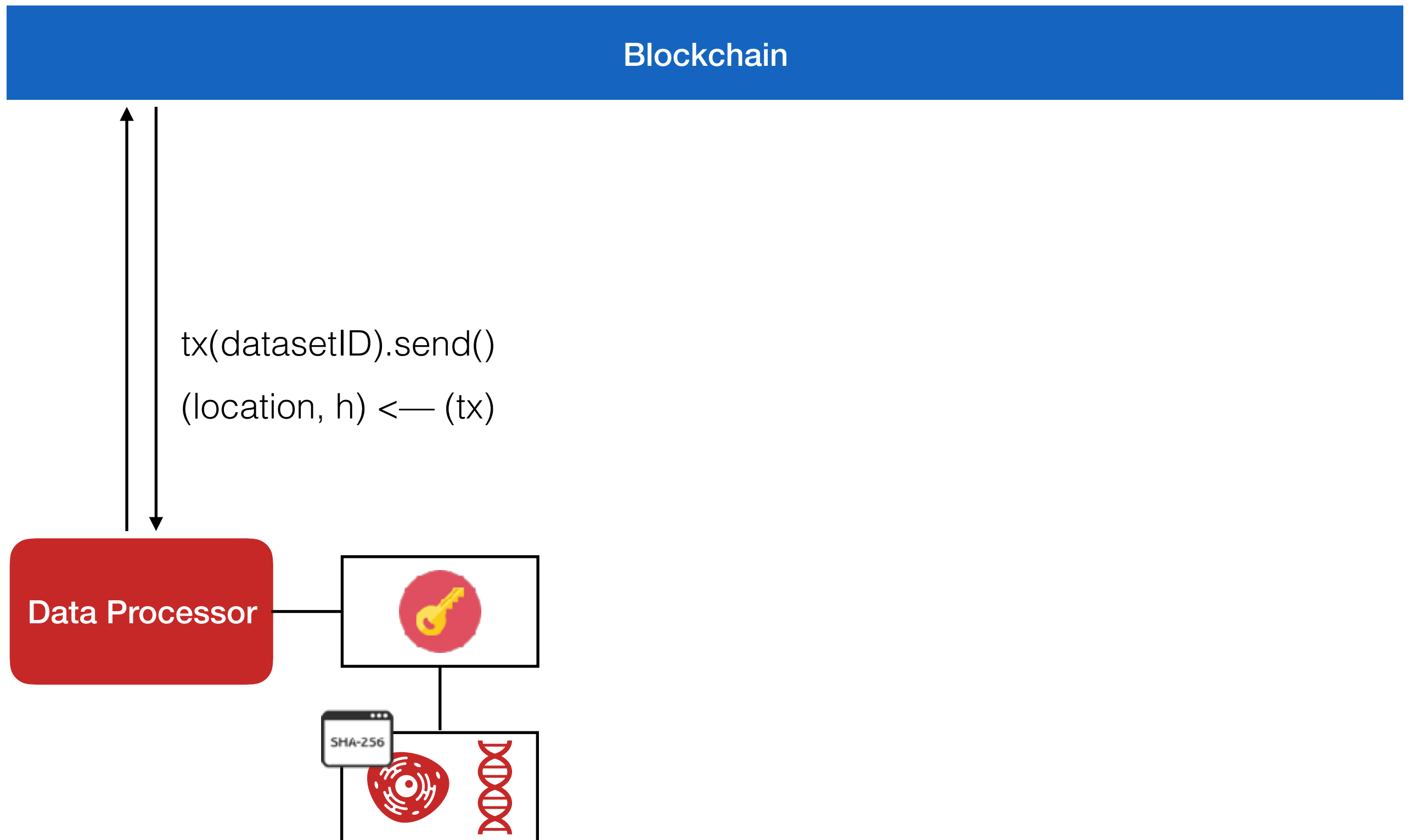
# Processing

Step 2:  
Decrypt key



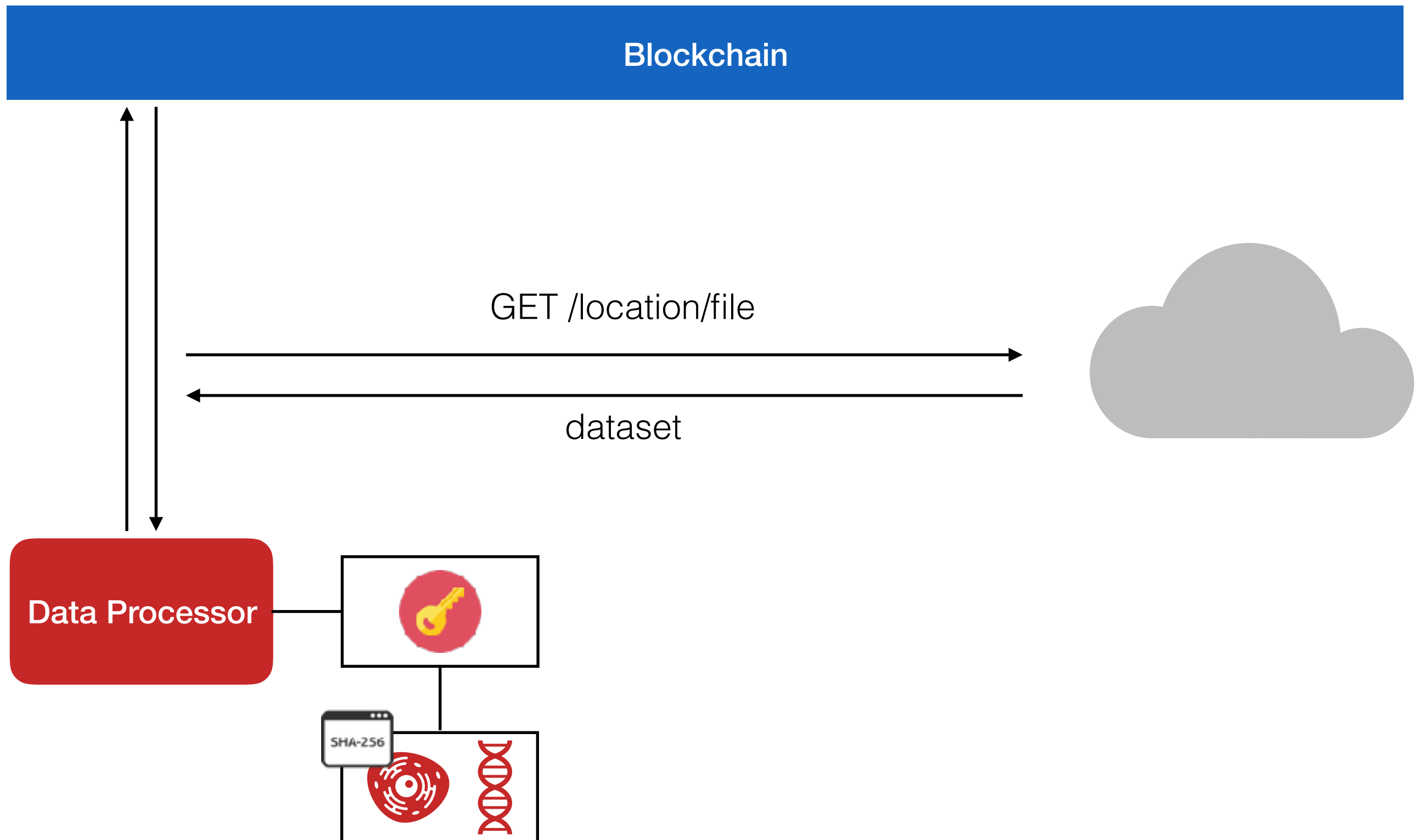
# Processing

Step 4:  
Get dataset info



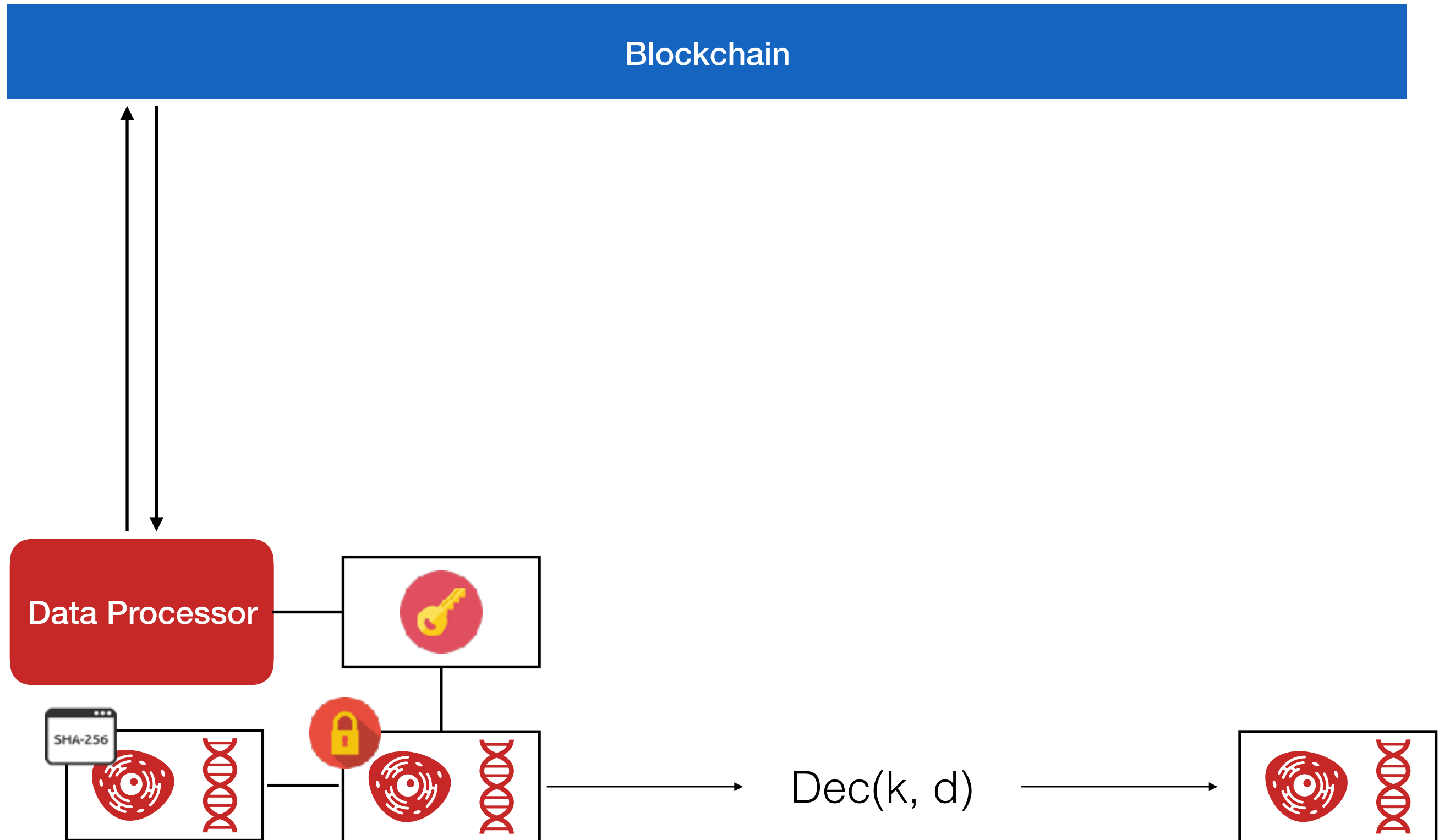
# Processing

Step 5:  
Get dataset



# Processing

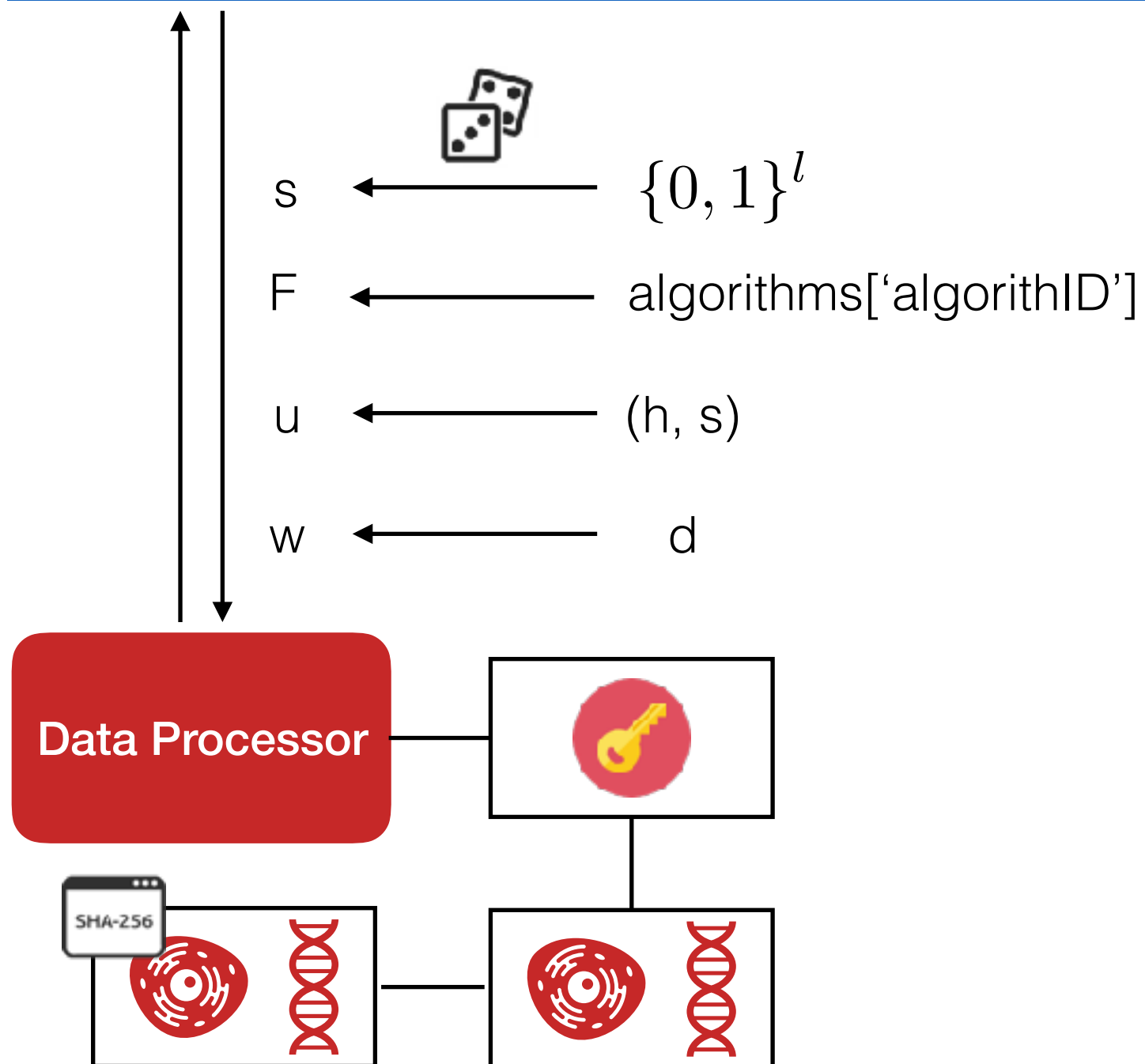
Step 6:  
Decrypt dataset



# Processing

Step 6:  
Proof

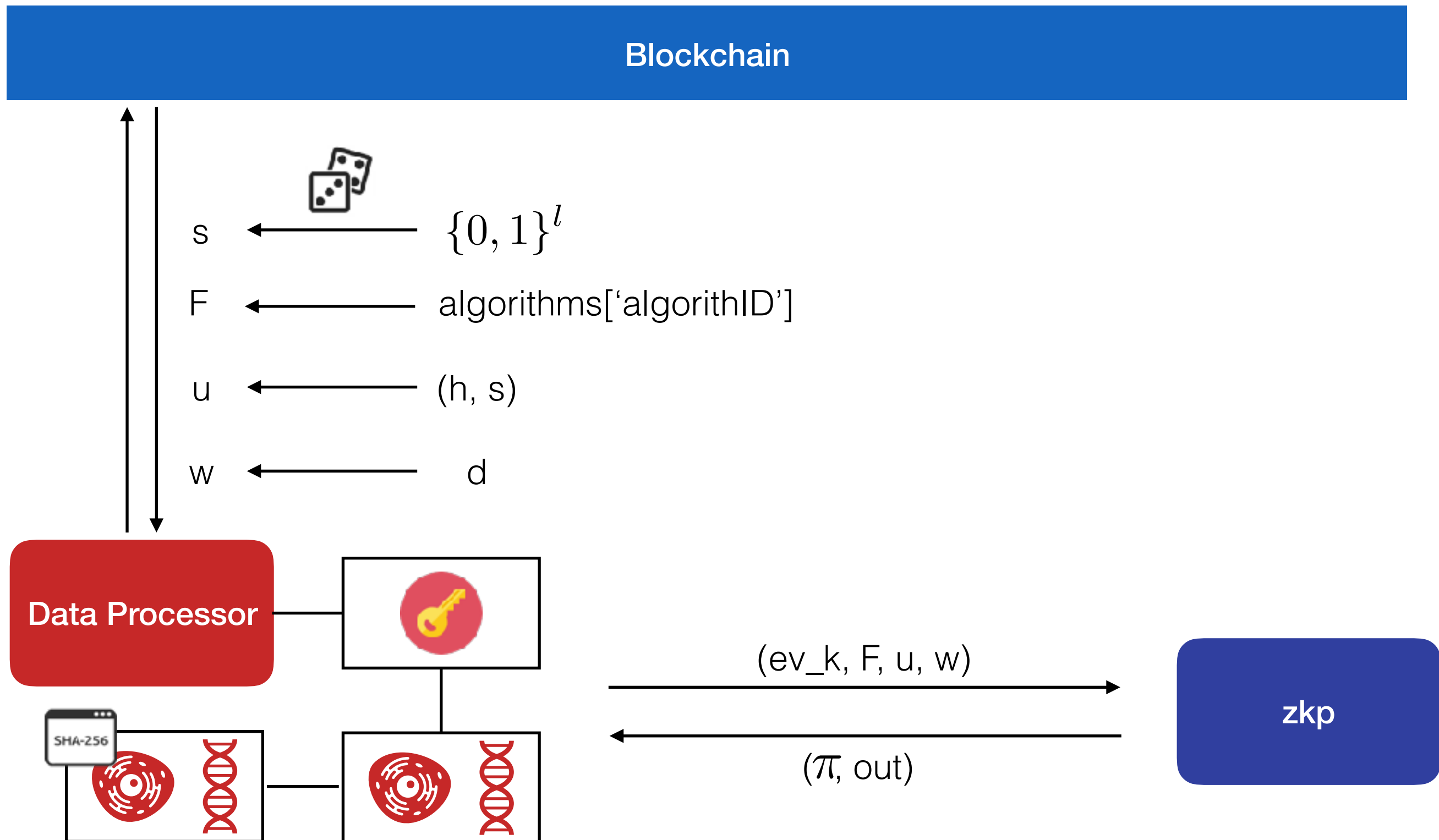
Blockchain





# Processing

Step 6:  
Proof



# Processing

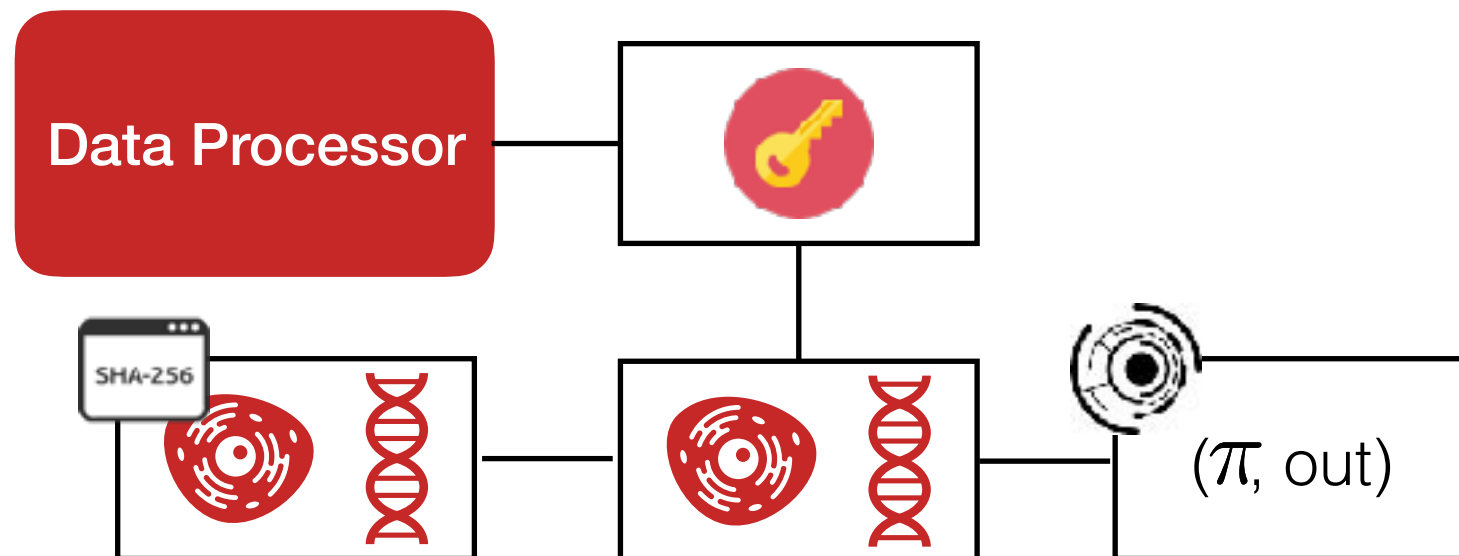
Step 6:  
Send proof



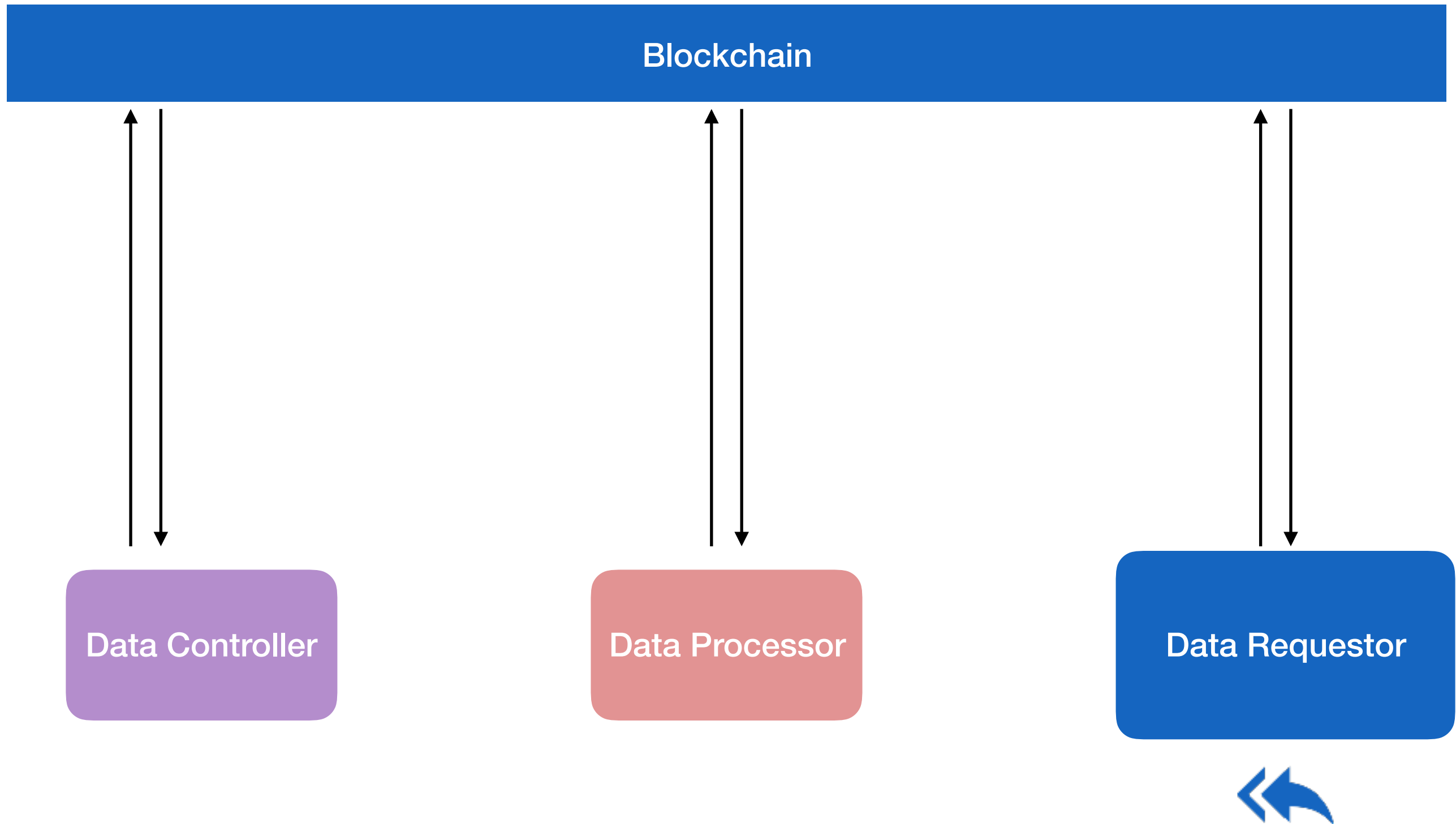
Blockchain

emit event: Completed(requestID)

$\text{tx}(\pi, \text{Enc}(\text{pk}, \text{out})).\text{send}()$



# Verification



# Verification

Step 1:  
Get proof

Blockchain

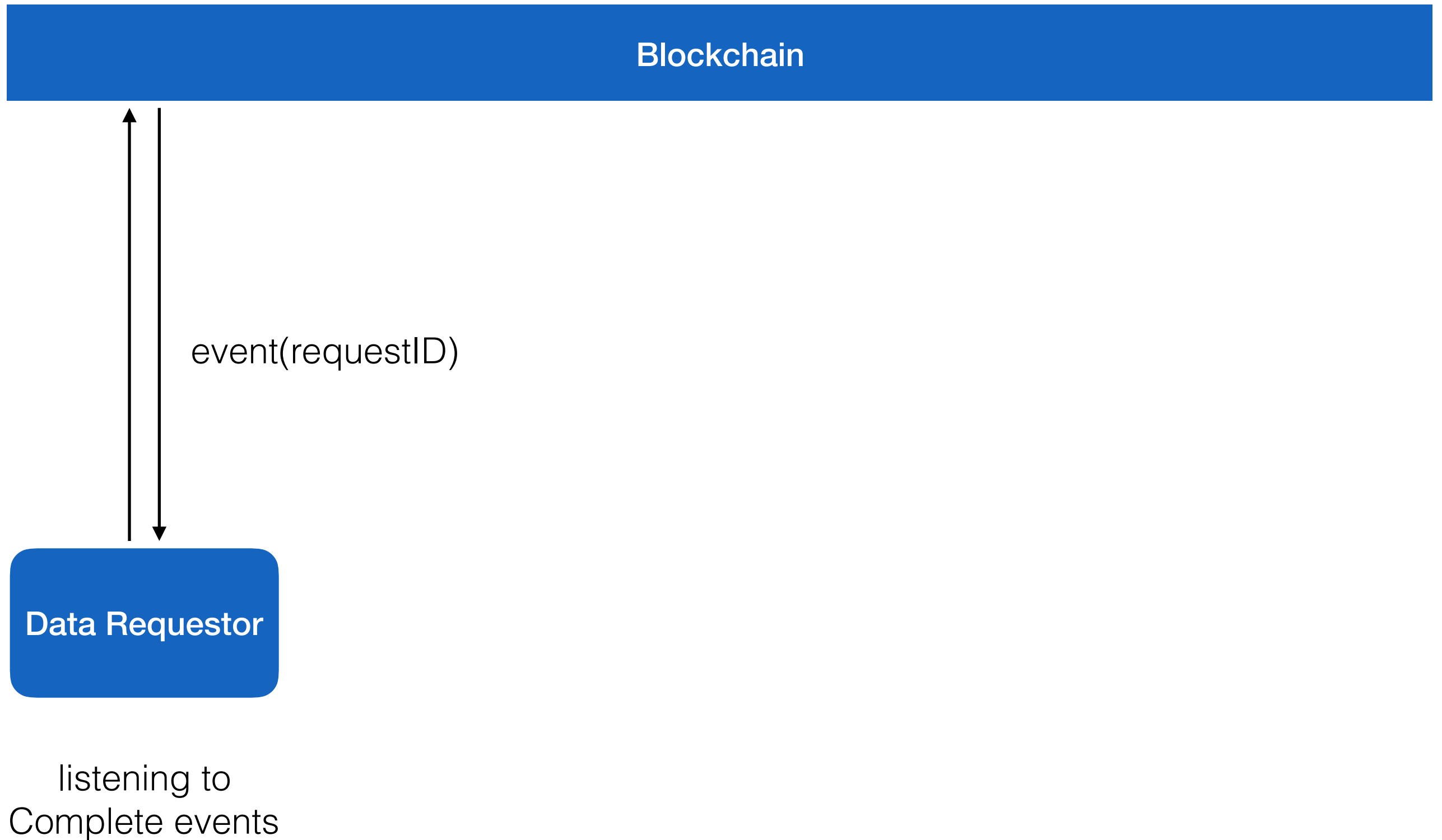


Data Requestor

listening to  
Complete events

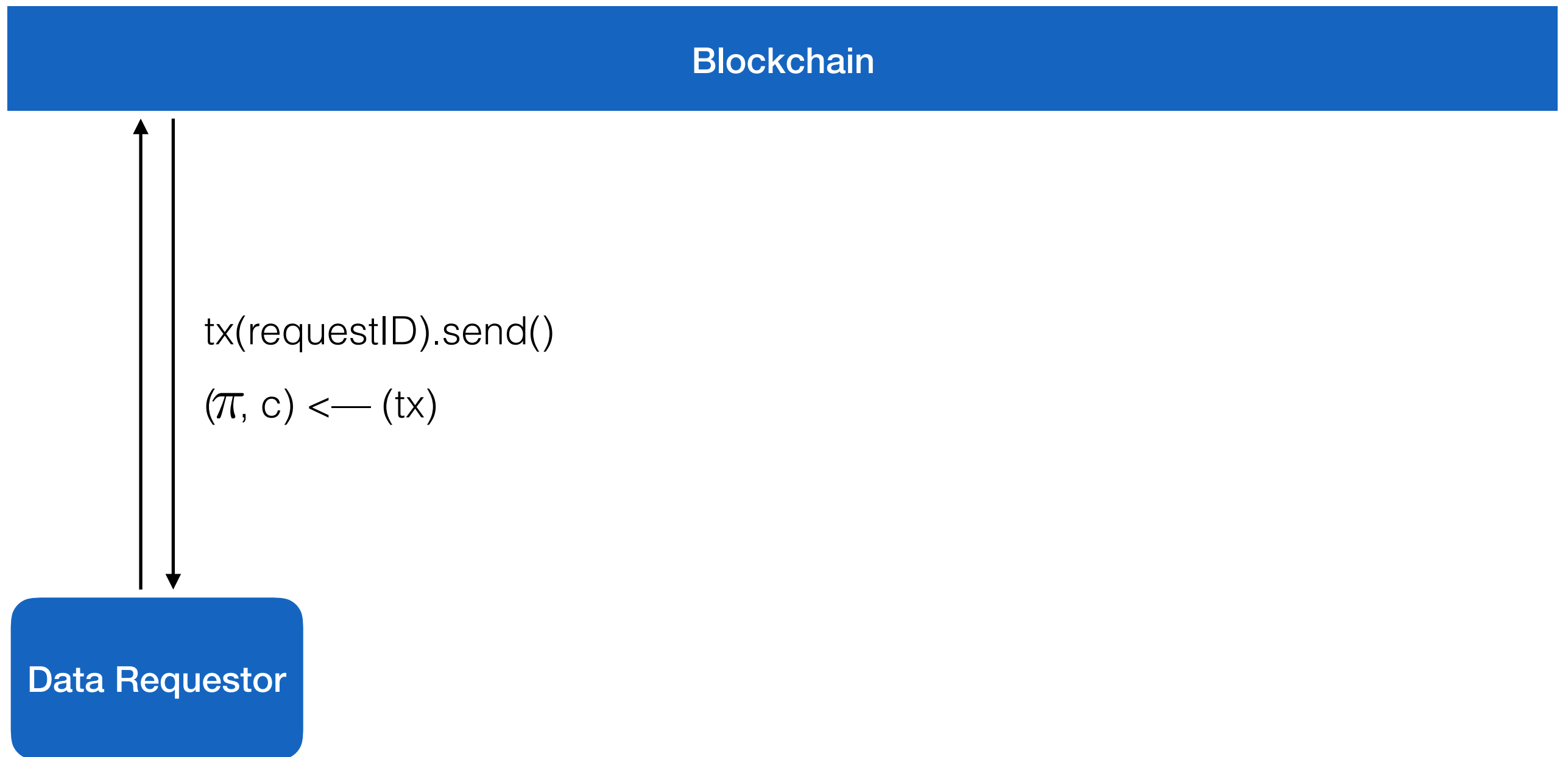
# Verification

Step 1:  
Get proof



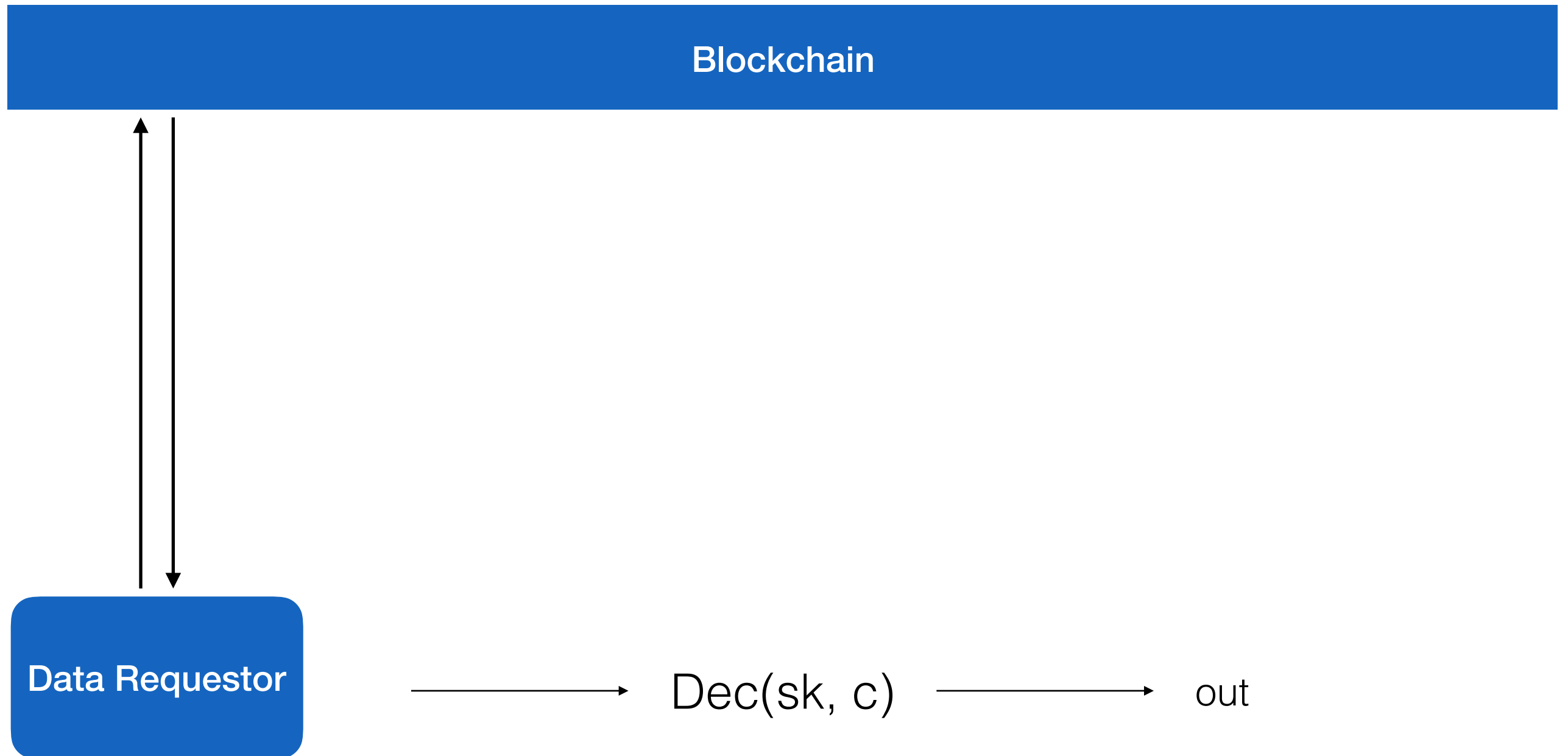
# Verification

Step 1:  
Get proof



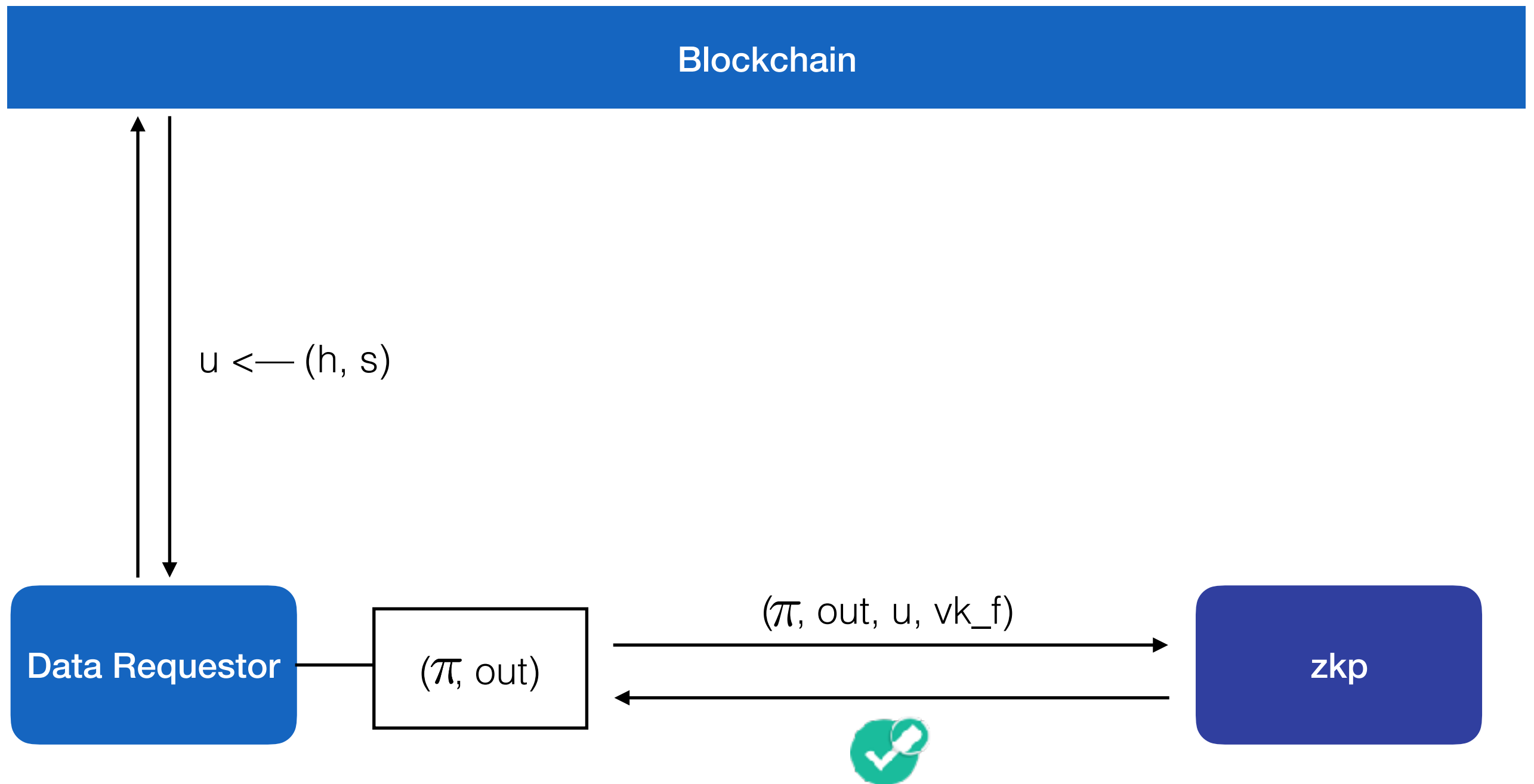
# Verification

Step 2:  
Decrypt results



# Verification

Step 3:  
Verify proof





# zkSNARKs [PHGR13]

Let **F** an outsource function, a public input **u**, a private input **w** and **y = F(u, w)**

A zero knowledge verifiable computation scheme consist of a set of three polynomial algorithms:

$$(ek_f, vk_f) \leftarrow \mathcal{G}(F, 1^\lambda)$$

$$(y, \pi) \leftarrow \text{Compute}(ek_f, u)$$

$$\{0, 1\} \leftarrow \text{Verify}(vk_f, u, y, \pi)$$

# Security assumptions

- Public key infrastructure (PKI)
  - Authenticates and verifies the data processor and the data controller
  - Perform the trusted setup for each algorithm

# Assumptions

- Data controller:
  - Integrity
  - Confidentiality
- Data processor:
  - Confidentiality

# Treat model



Data Processor



Data Controller



Data Requestor

# Treat model

- Fabricated dataset to manipulate results
- Collusion with data processor



Data Controller

# Treat model

- Fake computation
- Process different dataset
- Use different algorithm
- Fake results
- Expose dataset



Data Processor

# Treat model

- DDoS attacks



Data Requestor

# Proof

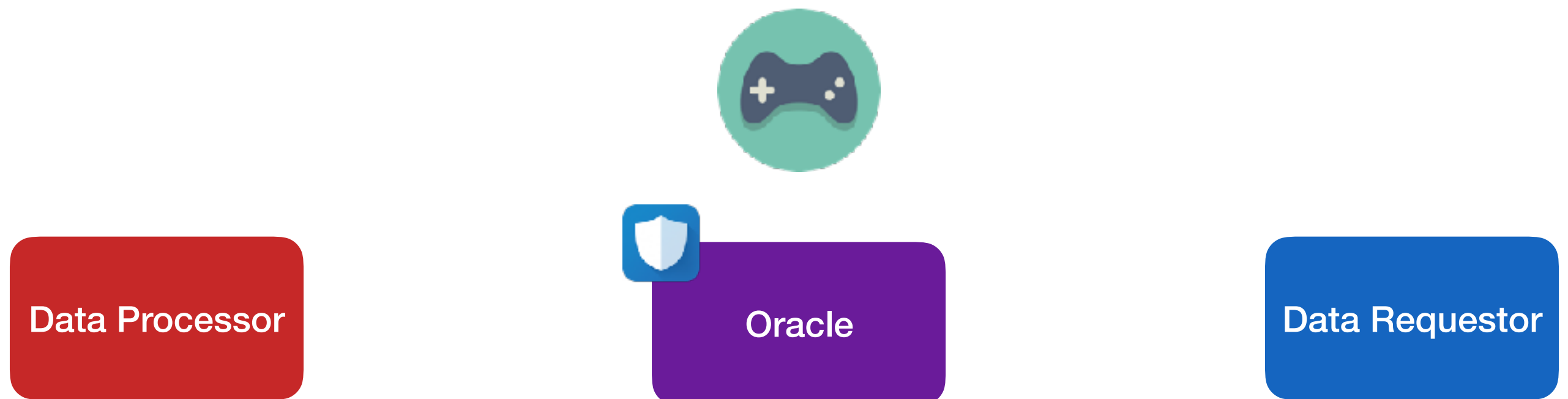
“I did the **correct** computation on the  
**requested** dataset and algorithm”



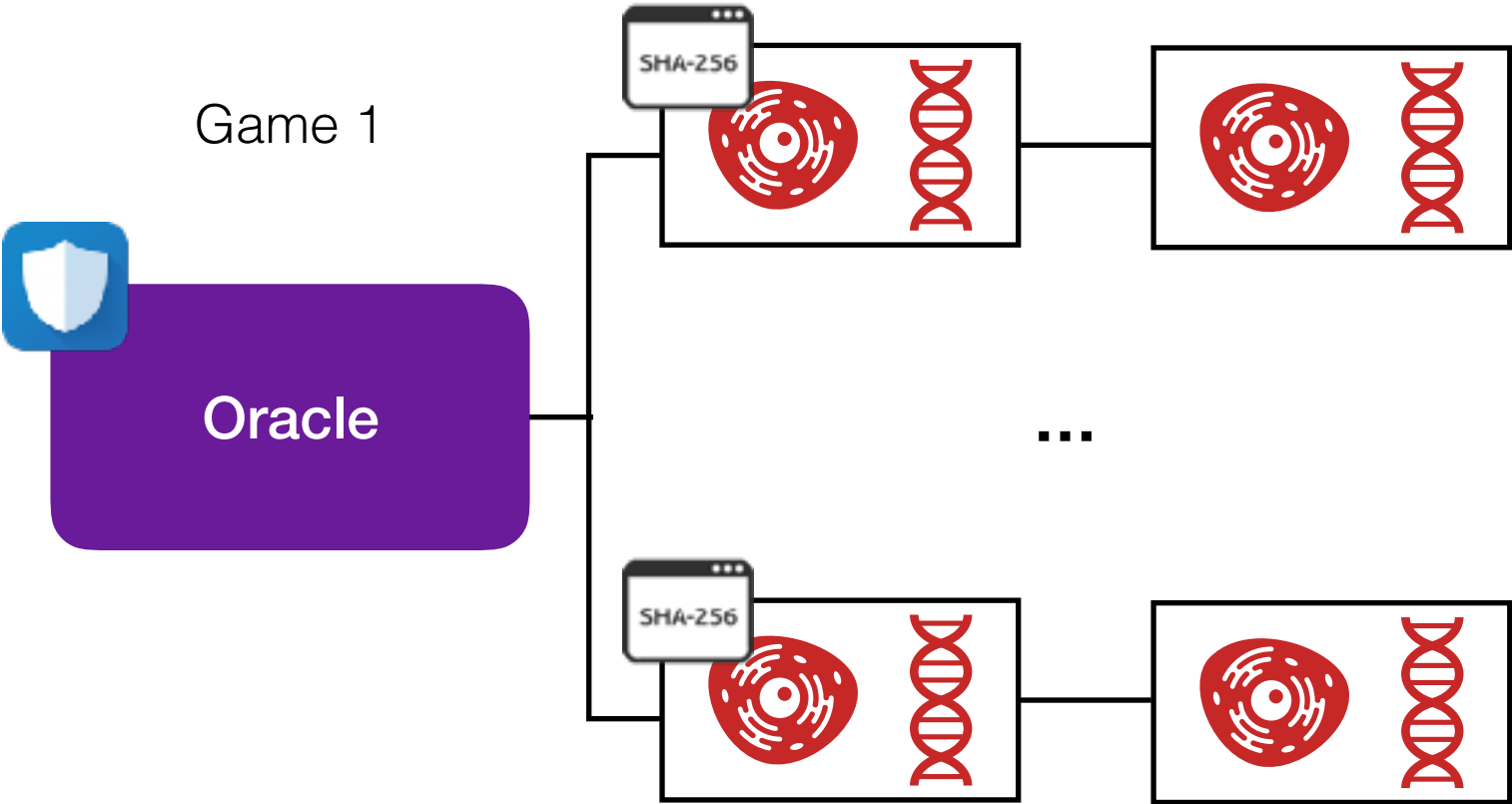
# Proof

Trusted Oracle:

- Hold a list of random datasets with corresponding digests
- The only owner of the datasets and digests
- Cannot lie about the content of the dataset neither of the digest
- No collusion with other participants



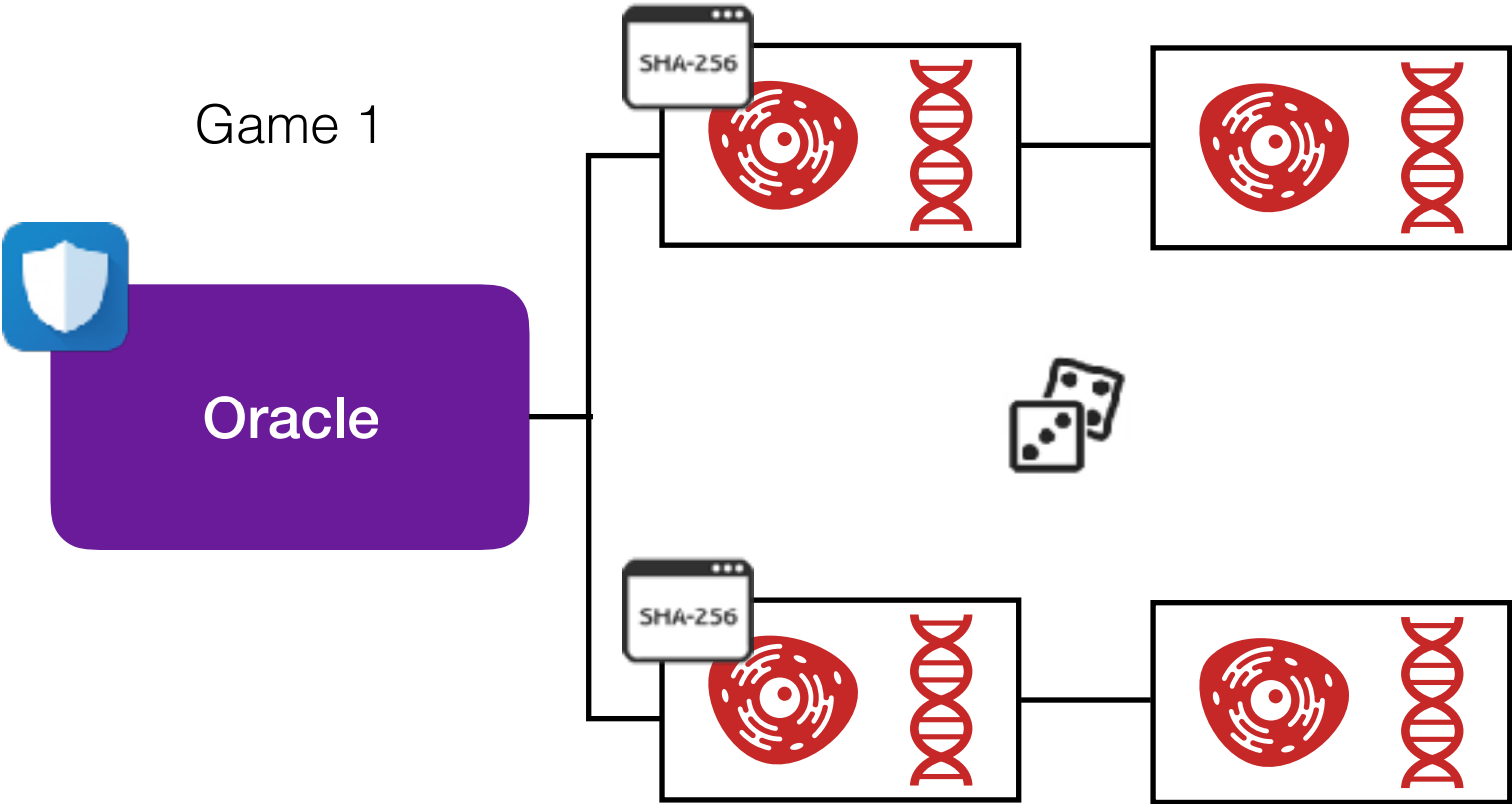
# Proof



Data Processor

Data Requestor

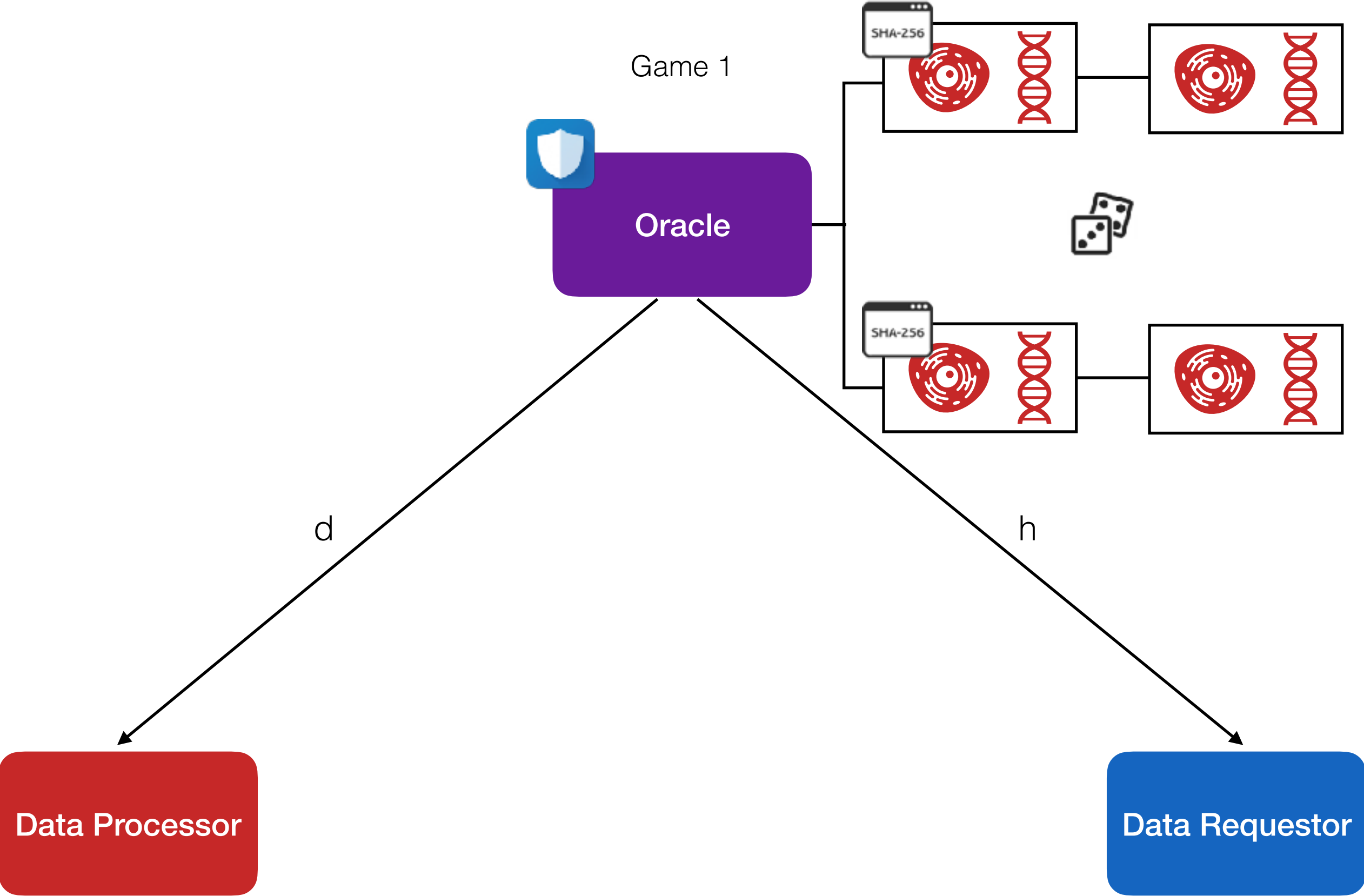
# Proof



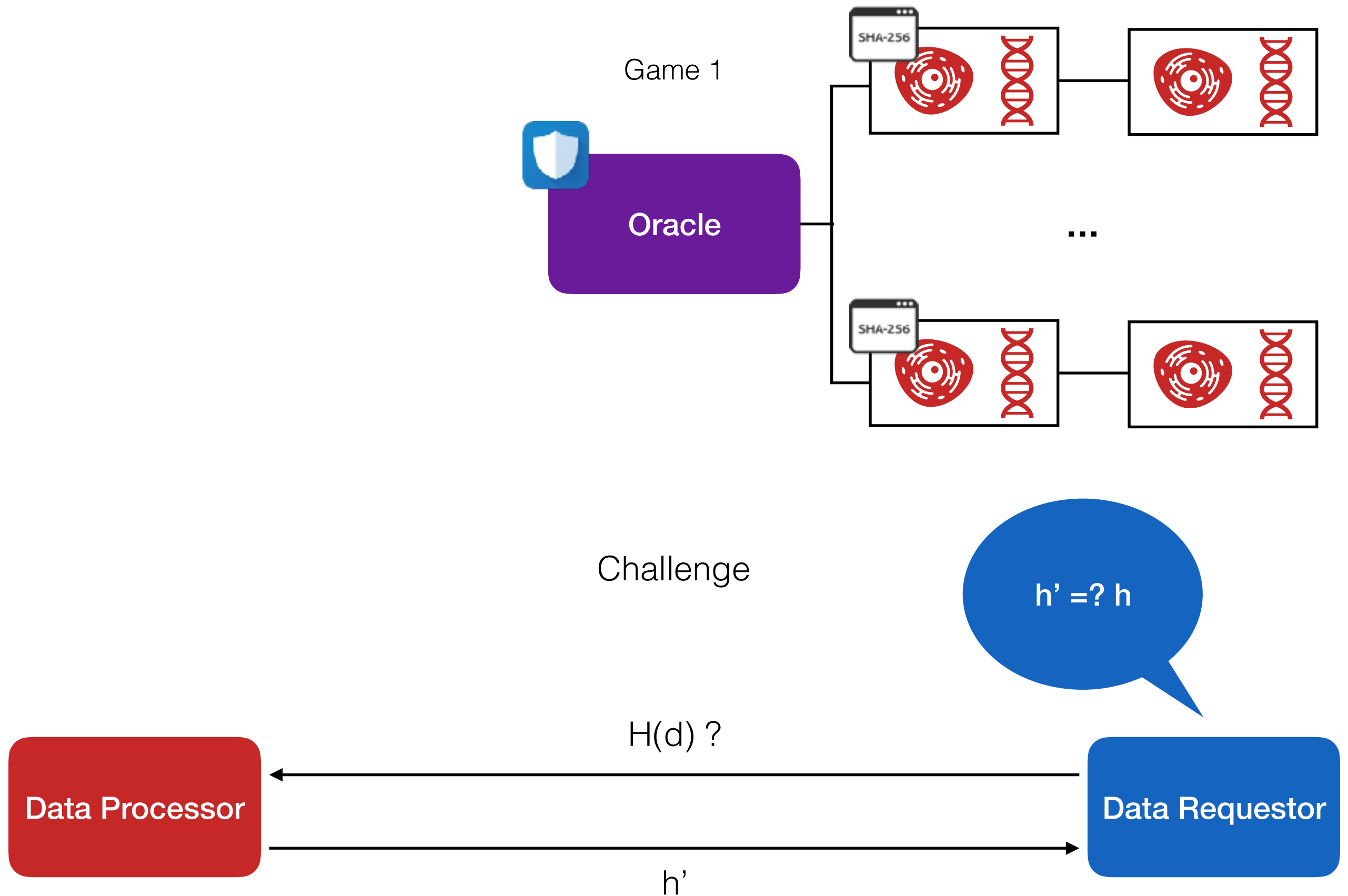
Data Processor

Data Requestor

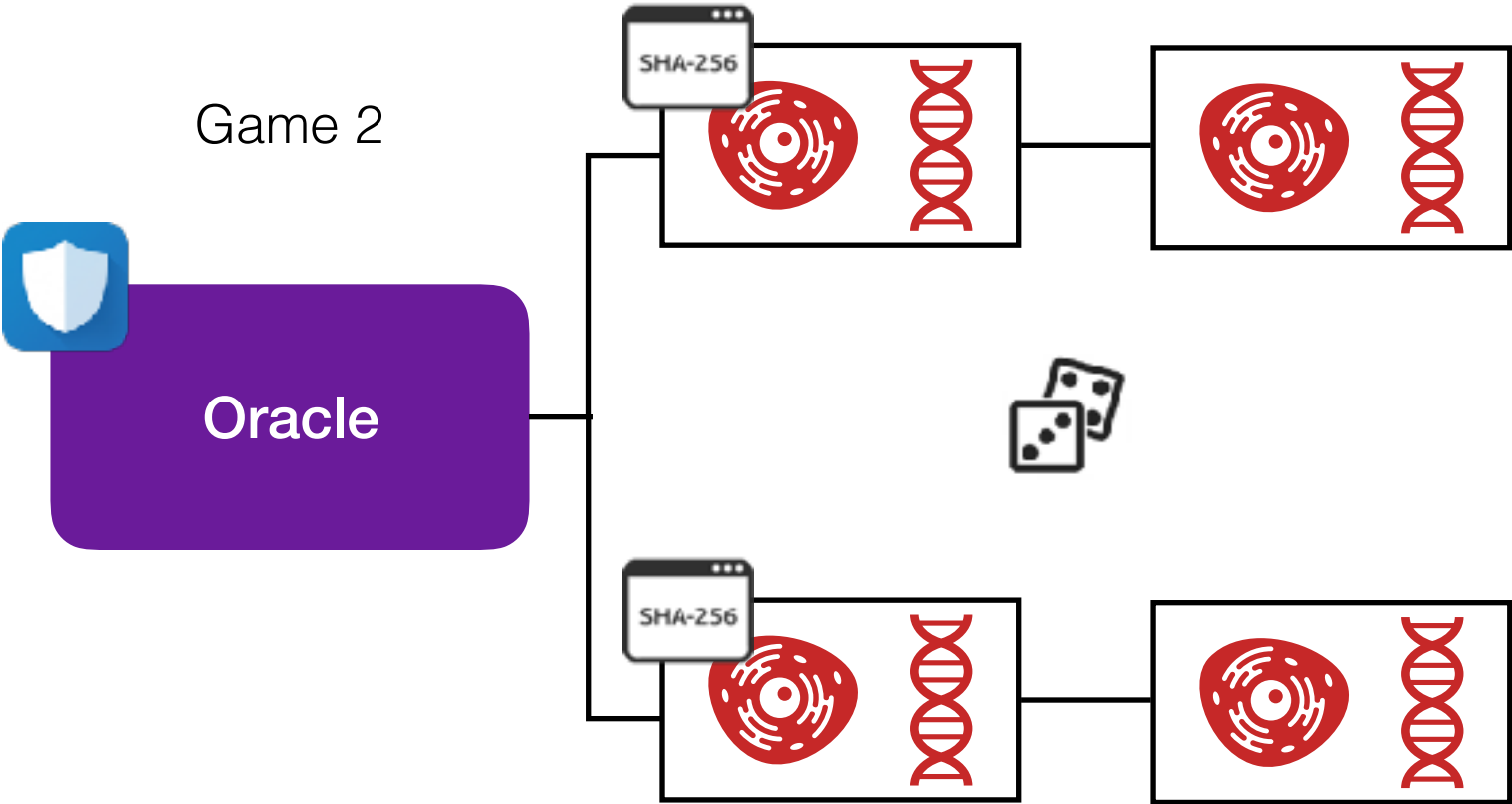
# Proof



# Proof



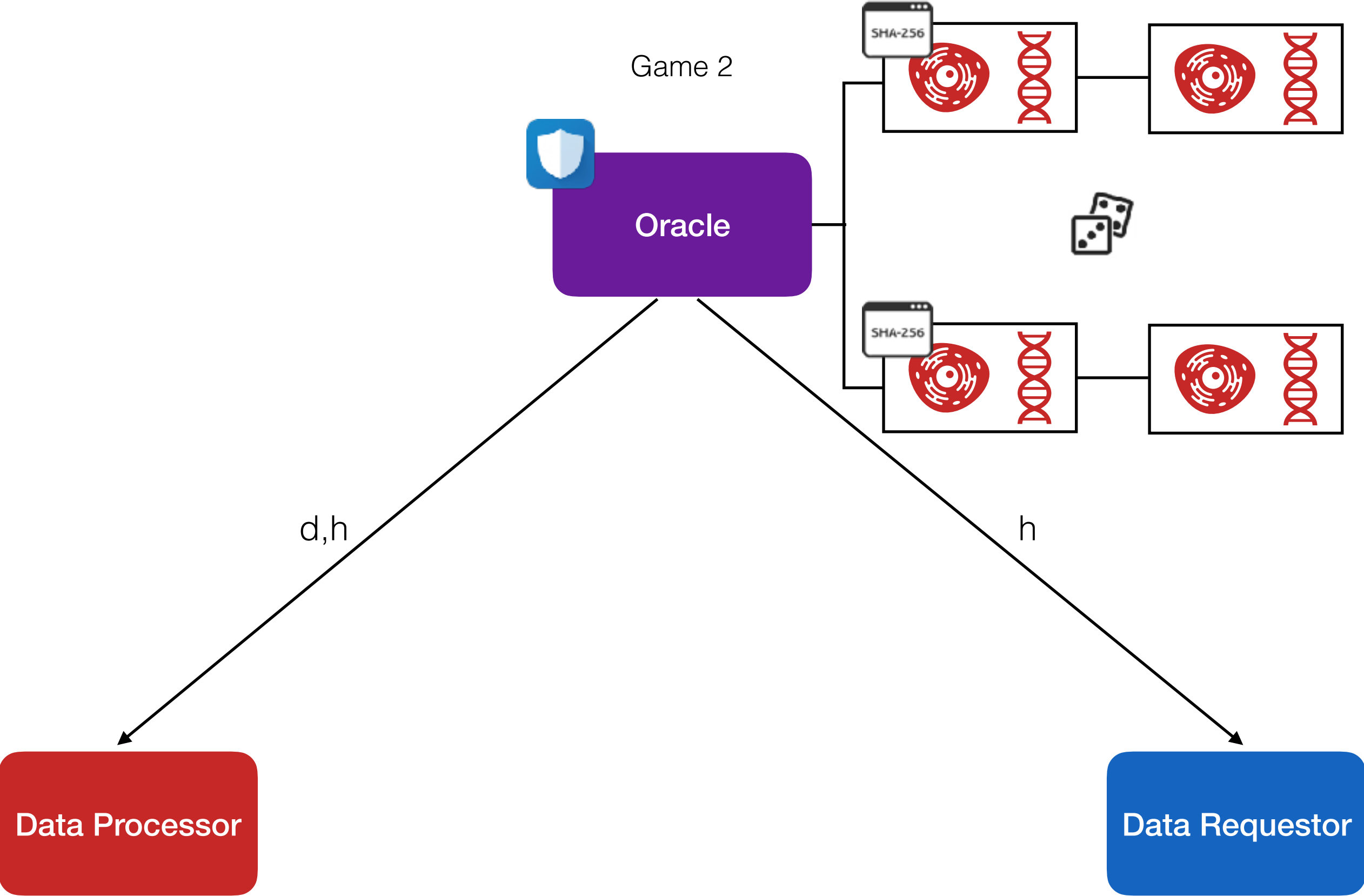
# Proof



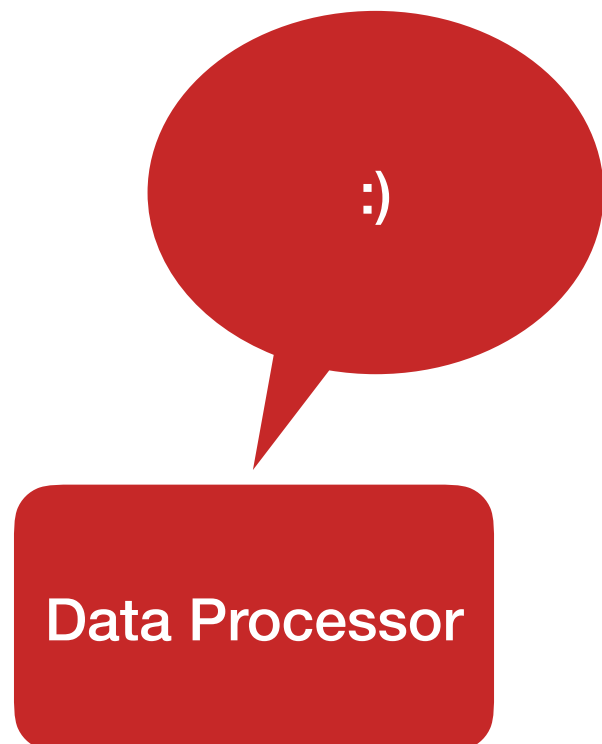
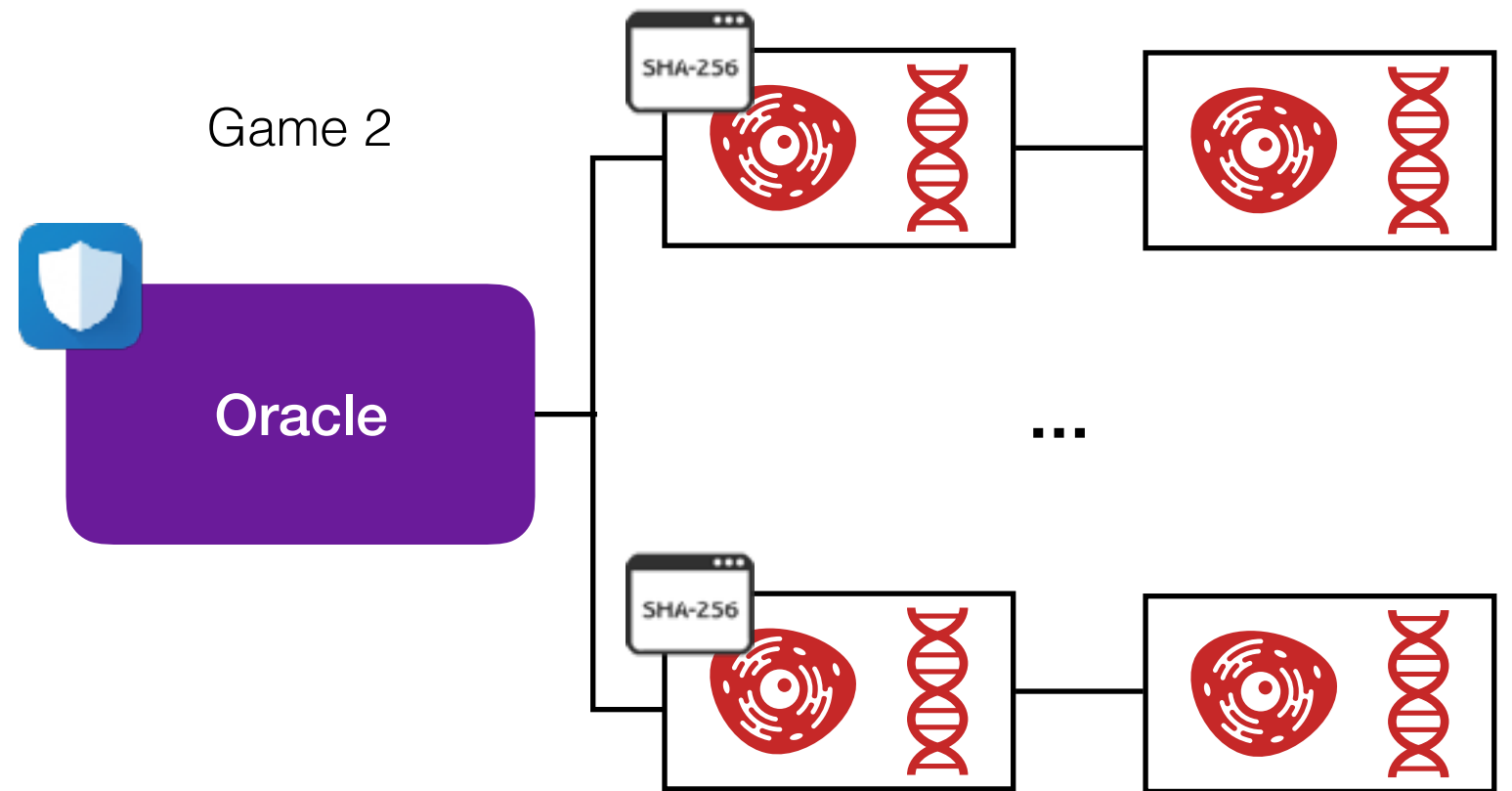
Data Processor

Data Requestor

# Proof

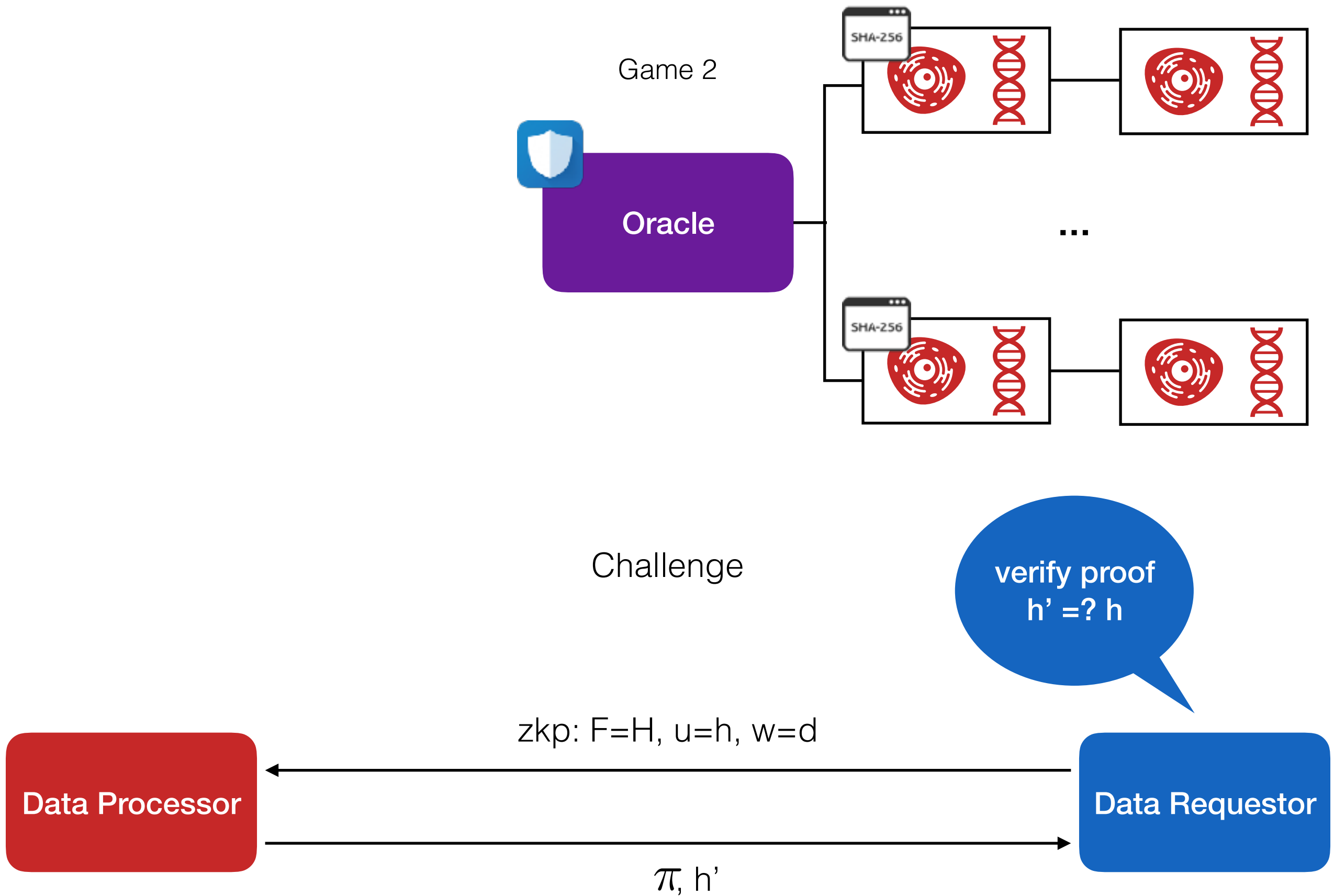


# Proof





# Proof



# Proof

- Ideally the controller could be the trusted oracle
- No security proof
- No indistinguishability assumption

Data Processor



Data Controller

Data Requestor

# Proof

“I know a dataset  $\mathbf{d}$  such as  $\mathbf{H}(\mathbf{d}) = \mathbf{h}$   
and I correctly compute  $\mathbf{F}(\mathbf{d}) = \mathbf{y}$ ”

# Proof

- The proof is public in the blockchain
- The results of the computation are encrypted with requestor's public key and in the blockchain
- But...

Data Processor

Data Requestor

# Proof

- The verification algorithm, verification key and the proof are public
- An adversary can brute-force and guess correctly the output

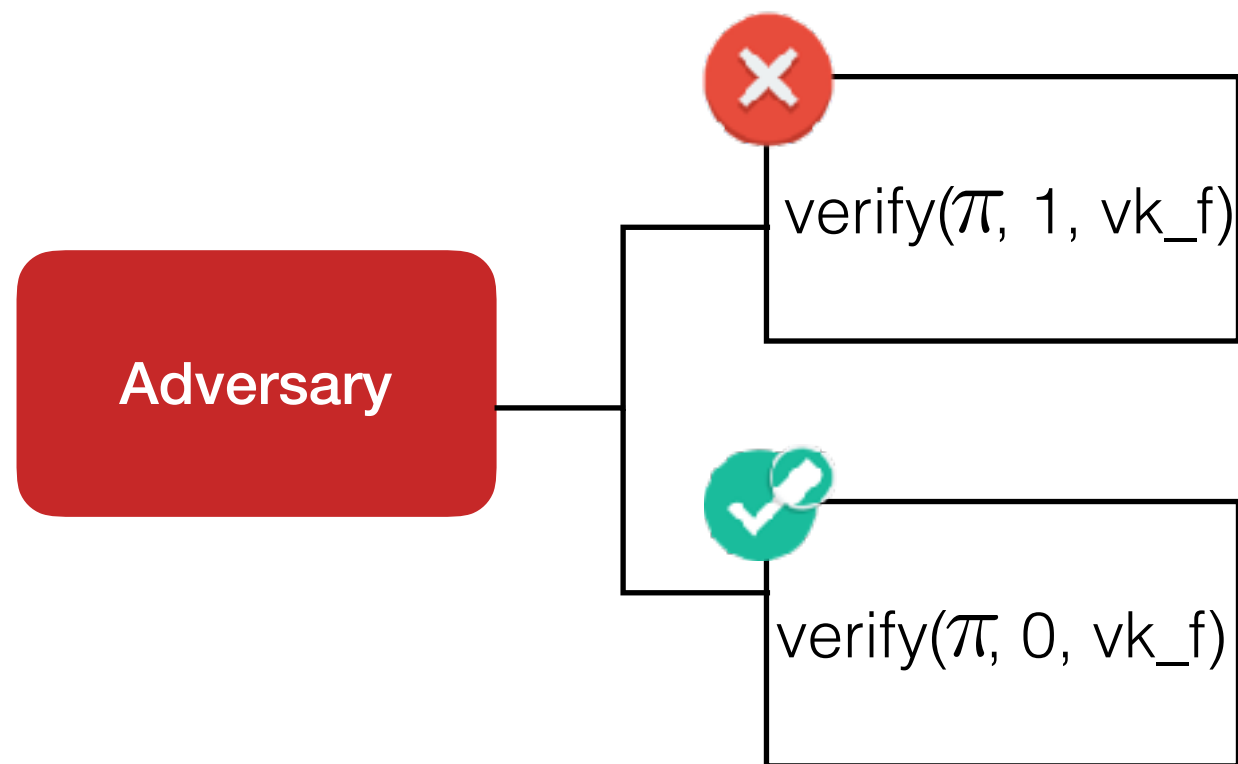
Adversary

Data Requestor

# Proof

Parity check

$$F : \{0, 1\}^n \rightarrow \{0, 1\}$$



# Proof

Solution

- Salt:  $s \xleftarrow{r} \{0, 1\}^l$
- Generated by the processor for each proof
- Public in the zkSNARK setting

# Proof

## Final Construction

Input:

- Public:  $h, s, ek_f$
- Private: dataset

Compute ( $F'$ ):

- $H(d) = h'$
- $F(d) = y$

Output:

- $(\pi, h', y, s)$



Evaluation

# Implementation

- Ethereum
- Smart contracts
- RESTful API
- DApp
- CLI
- Blockchain, Storage, Crypto libraries
- Controller, Processor, Requestor nodes
- ZKP ecosystem

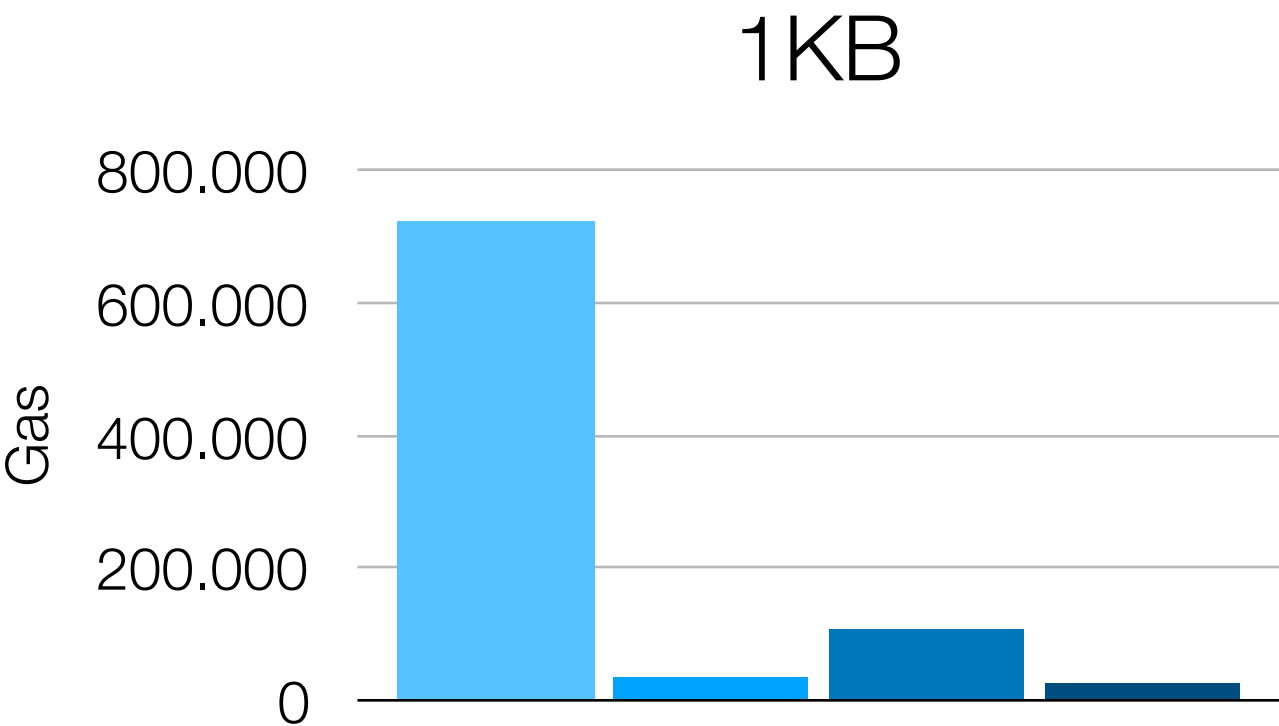
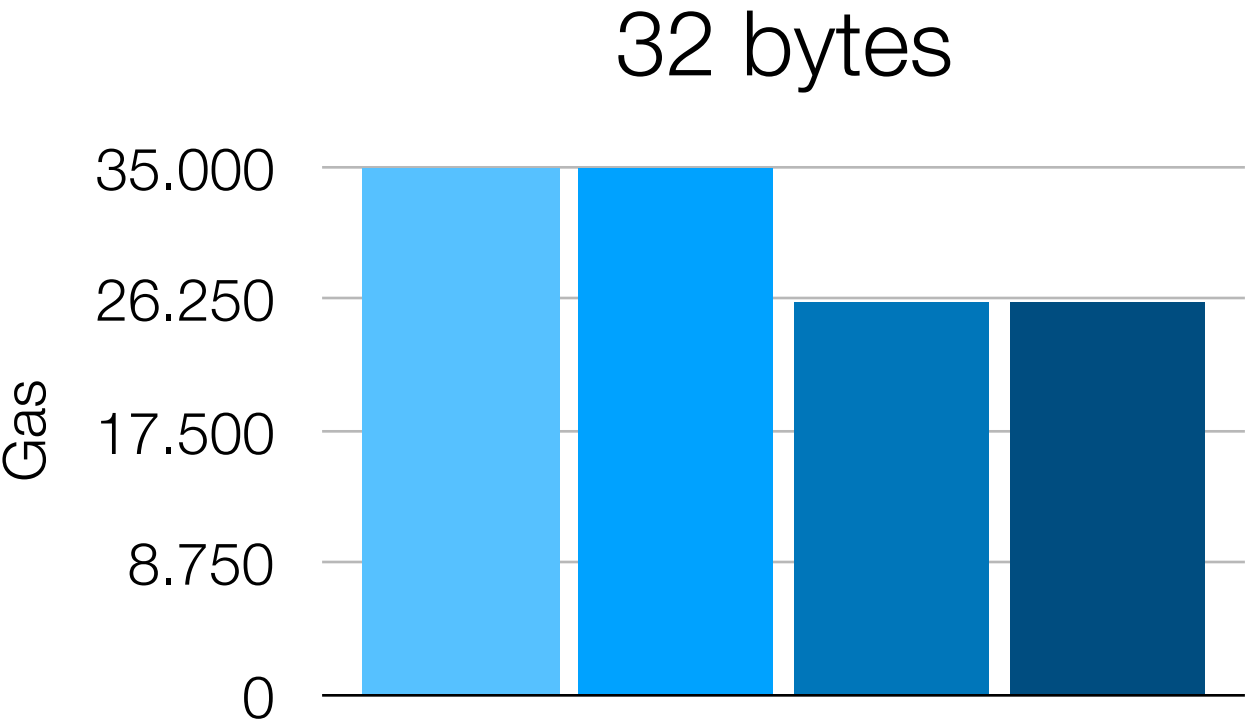
# Evaluation

*How much it cost to store data ?*

Size	Gas	Cost
32 bytes	20.000	\$0,037
1KB	724.664	\$1.357
1MB	697.325.562	\$1,305.393
10MB	~7.000.000.000	~\$13,104
100MB	~70.000.000.000	~\$131,040
1GB	~700.000.000.000	~\$13,104,000

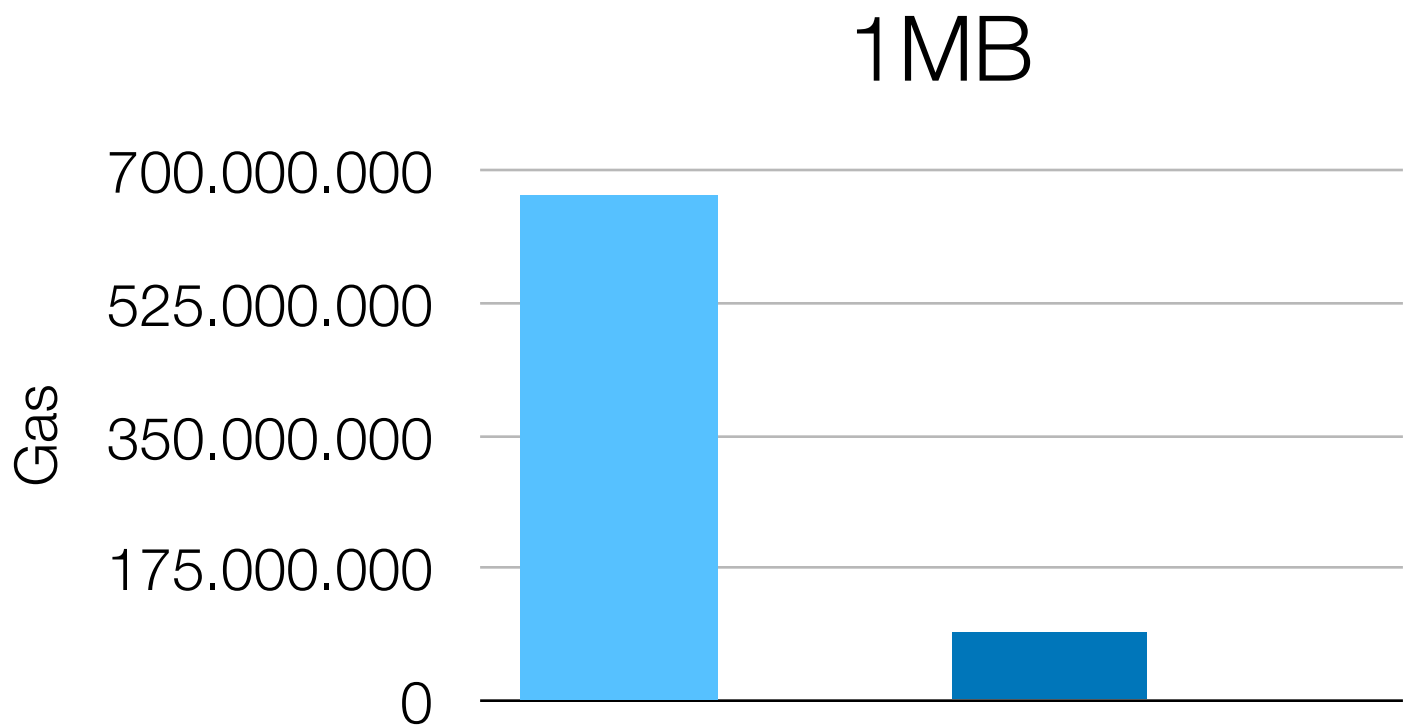
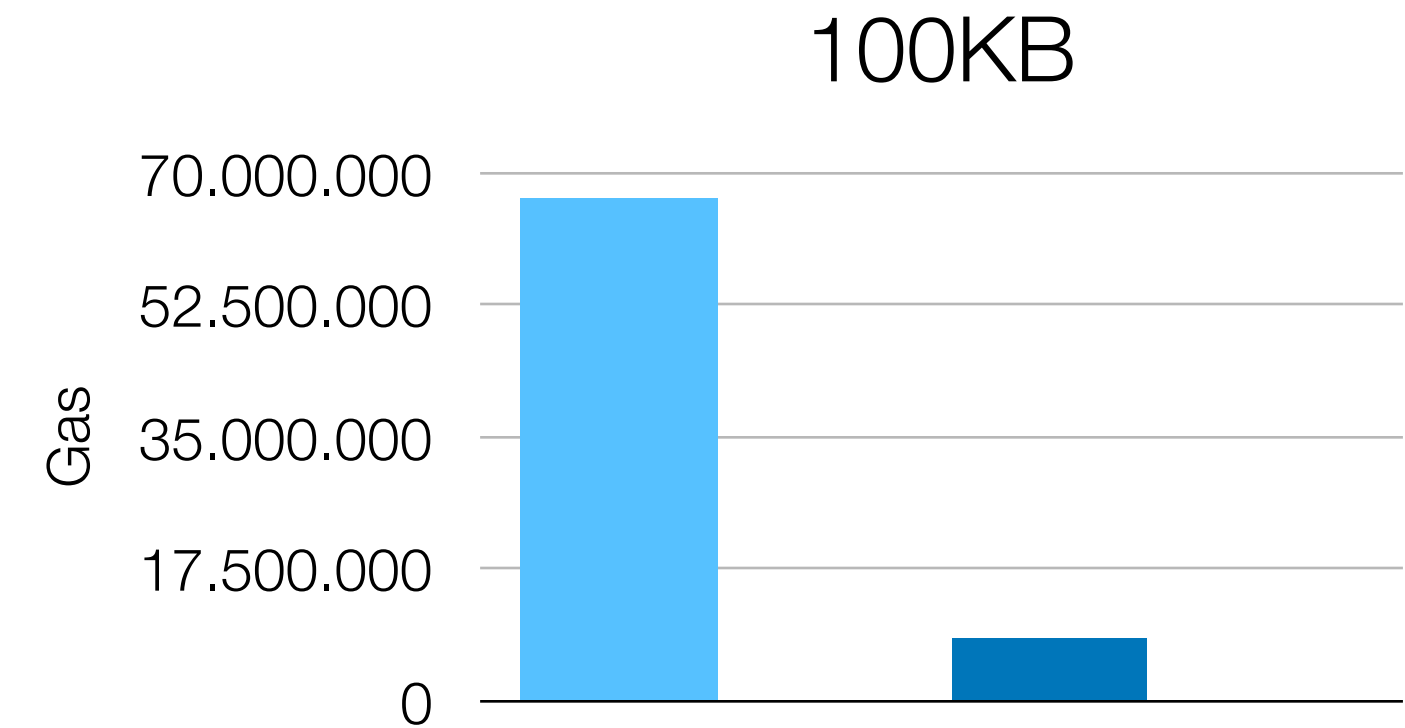
ETH Price: \$942.99 (Feb 18, 2018) - Gas Price: 2 Gwei

# Evaluation: Storage costs



ETH Price: \$942.99 (Feb 18, 2018) - Gas Price: 2 Gwei

# Evaluation: Storage costs



ETH Price: \$942.99 (Feb 18, 2018) - Gas Price: 2 Gwei

# Evaluation: Storage costs

*How much it cost to use the system ?*

Type	Gas	Cost
App deployment	3.004.253	\$5.66
Register	233.487	\$0.44
Request	89.206	\$0.15
Process	83.274	\$0.15
Register entity	25.279	\$0.04
Proof	100.248	\$0.18

ETH Price: \$942.99 (Feb 18, 2018) - Gas Price: 2 Gwei

# Evaluation: ZKP

	<b>Setup (s)</b>	<b>Compute (s)</b>	<b>Verify (ms)</b>	<b>Eval key (MB)</b>	<b>Ver key (KB)</b>	<b>Proof (B)</b>
Sum	10	8	27	0,144	100	304
Count	15	9	37	0,036	100	304
Min / Max	29	14	49	17	100	304
Median	1700	780	12	64	100	304
Mean	16	8	10	0,82	100	304

1000 values

# Future Work

- MPC
- Fees
- Reputation System
- Privacy Preserving Algorithms
- New Ideas ?



# Related Work

- Enigma
- MedRec
- AD-SNARKs
- Zokrates

Questions ?

