Disciplina: Blockchain for Education

Kirill Kuvshinov[1], Jonn Mostovoy[2], and Ilya Nikiforov[1]

[1]Teach Me Please, https://teachmeplease.com
[2]Serokell, https://serokell.io

Version 0.5.1
December 30, 2017

Abstract

In this paper we analyze the main issues that arise from storing educational records in blockchain and propose the architecture of the Disciplina platform – a domain-specific blockchain implementation. The platform is designed to act as a decentralized ledger, with special regard for privacy and mechanisms of data disclosure. We present an overview of the main entities, their roles and incentives to support the network. Please note that the project is a work-in-progress and the descriptions provided are subject to change.

1 Introduction

Recent advances in blockchain technology and decentralized consensus systems open up new possibilities for building untamperable domain-specific ledgers with no central authority. Since the launch of Bitcoin [**?**] blockchains had been primarily used as a mechanism for value transfers. With the growth of the Ethereum platform [**?**], the community realized that by using a chain of blocks and consensus rules one can not only store value and track its movement, but, more generally, store some state and enforce conditions upon which this state can be modified.

Bitcoin, Ethereum and other permissionless blockchains were developed with the assumption that everyone is free to join the network and validate transactions, that are public. However, the industry often requires privacy, and thus the permissive solutions with private ledgers came to exist. These solutions include Tendermint [**?**], Hyperledger [**?**], Kadena [**?**] and others.

The increased interest and the variety of the blockchain technologies lead to the growth of their application domains. The idea of storing educational records in the blockchain has been circulating in the press and academic papers for several years. For example, [**?**] and [**?**] focus on the online education and propose to create a system based on the educational smart contracts in a public ledger. Recently, Sony announced a project that aims at incorporating educational records in a permissioned blockchain based on Hyperledger [**?**]. The ledger is going to be shared between major offline educational institutes.

The main issue these solutions have in common is that they target a certain subset of ways people get knowledge. We propose a more general approach that would unite the records of large universities, small institutes, schools and online educational platforms to form a publicly verifiable chain. Contrary to the solutions like Ethereum, we do not aim at proposing a programmable blockchain that fits all the possible applications. Rather, we believe, that we should harness all the latest knowledge that emerged in the last few years in the fields of consensus protocols, authenticated data structures and distributed computations to offer a new domain-specific ledger. In this paper we introduce Disciplina — the platform based on blockchain technology that aims to transform the way educational records are generated, stored and accessed.

2 Architecture overview

Due to the nature of the platform, it has to operate on sensitive data, such as courses, assignments, solutions and grades. Permissionless blockchains, like Ethereum or EOS, would require disclosing this data to the public, whereas the permissive ones, like Hyperledger, lack public verifiability. Our

architecture splits the blockchain into two layers: the private layer contains sensitive data, and the public one contains the information necessary to validate the integrity and authenticity of the private blocks. The key entities of the proposed blockchain architecture are presented in Figure 8.

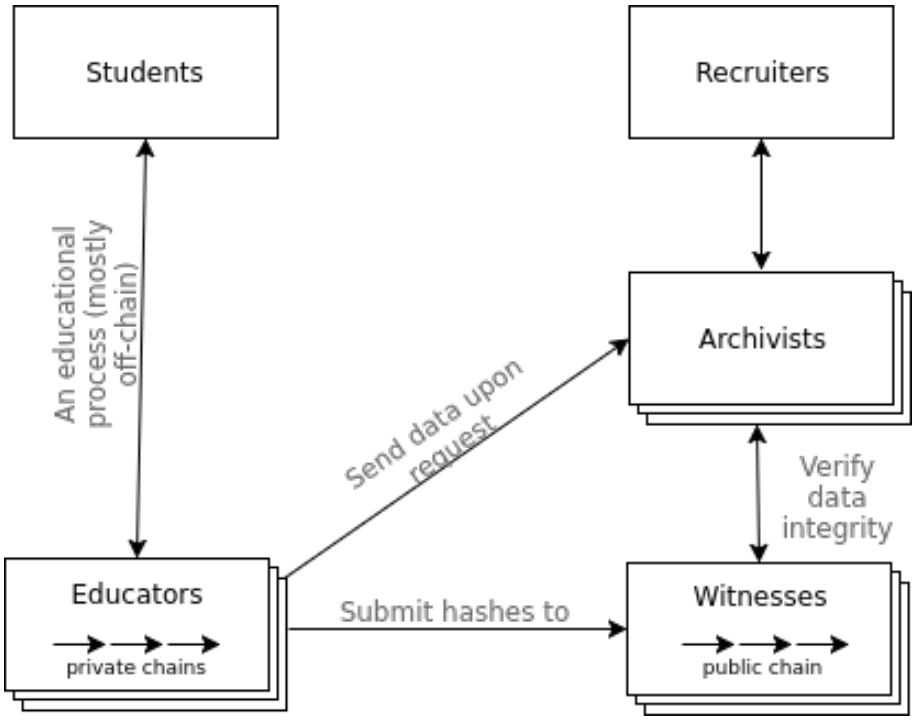

Figure 1: Key entities of the Disciplina platform

The private layer is maintained by each Educator independently of others. Educators can be either large educational institutes, capable of running their own nodes, or some trusted party that runs the chain for the self-employed teachers and small institutions. This layer contains the personalized information on the interactions between the students and the Educator. All the interactions, such as receiving an assignment, submitting solutions, or being graded, are treated as transactions in the private chain.

Students get access to the platform through web and mobile applications. Using the applications they choose Educators, enroll in courses, get assignments and submit solutions. The scores and the criteria of whether the Student has finished the course successfully are determined by the Educator. The education process from the platform's perspective is as follows:

1. A Student chooses an Educator and a course that she wants to enroll in.
2. If the course is offered on a pre-paid basis, the Student uses her app to pay the fee.
3. During the course, the Educator provides assignments that the Student has to complete in order to get the score.
4. The Student acquires the assignment, completes it and sends the signed solution back to the Educator (communication between the Student and the Educator happens off-chain).
5. The Educator then stores the solution locally, grades it, and transfers the score with the hash of the solution to the blockchain.
6. Upon the completion of the course, the Student acquires a final score based on the scores she got for her assignments. This final score is also added to the Educator's chain.

Making the Educators' chains private opens the possibility for Educators to tamper with the data in their chains. To overcome this issue and make the private transactions publicly verifiable, we introduce the second, public, layer of the blockchain. The public part of the network consists of Witnesses – the special entities that witness the fact that a private block was produced by an Educator. They do so by writing the authentication information of a private block into the public

chain, which is used in the future by an arbitrary Verifier to substantiate a proof of transaction inclusion given to it by a Student or an Educator. Witnesses also process public information issued by the Educators, such as announcements that an Educator has started or stopped offering a course in a particular discipline. The Witnesses agree on which public blocks are valid using the specified consensus rules.

Archivists are entities that provide an interface for the Recruiters to communicate with the platform. The Archivists act like a bridge between the Recruiters and the Educators: they choose the institutes that are relevant for the particular request, orchestrate the data disclosure and provide the evidence on the validity of that data. They obtain this evidence by communicating with Witnesses and comparing the headers of the private data blocks with the headers stored in the public Witnesses' chain.

3 Implementation choices

In this section we describe the proposed architecture in more detail. We present the excerpt on the internal structure of both public and private chains and the reasoning behind these choices.

In order to deduce the internal structure of our system, we first analyze its use-cases. The overview of the education process is given in Section 2. The communication between the Student and the Educator is saved as transactions in the private chain. However, the implementation details of this chain mostly depend on the data disclosure process.

We will start from analyzing this process and determining the main issues that arise from the need to disclose and verify the validity of the private blocks. Then we will propose the structure of the private and public blocks that addresses these issues.

3.1 Anonymity and certification

The permissionless nature of our public chain leads to the ability for malevolent students to create educational institutes in order to get the scores for the courses they did not attend. Moreover, the knowledge students actually get by completing the course, and the conditions upon which the course is considered completed, vary significantly between the educational institutions.

These issues currently can not be solved solely on the protocol level: they require an external source of information to determine the physical existence and the reputation of an Educator. Although we leave the public chain open for the Educators to submit their private block headers, we propose to add a separate layer of reputation and trust on top of the protocol.

We do so by introducing the Archivists – the entities that join the network with the approval from another Archivist. Their main role is to add a trust level above just the raw protocol: they store the certificates of the Educational institutes and have the right to revoke those certificates. Furthermore, the Archivists are the entities responsible for determining the ratings of the particular Educators. The Archivists base their rating on the off-chain sources of information and gain authority for providing valid ratings and performing all the necessary compliance procedures for the Educators.

While in theory the Recruiters can query the Educators directly and initiate a data disclosure request, they are generally discouraged to do so. We expect the Recruiters to be willing to pay extra fees to the Archivists for certificate validation and ranking of the Students depending on the Educators' ratings and other factors upon request. We describe the data disclosure process in detail in section 3.7

3.2 Activity Type Graph

When a Recruiter makes a request to one of the Archivists, the Archivist has to somehow choose the relevant Educators. Moreover, when an Educator discloses the data, it has to provide as minimal set of entries as possible. This set has to be verifiable, which means that the Educator provides the proof of the data validity along with the data being disclosed.

In order to achieve these goals, we divide the data that the Educators store into atomic Activity Types. Each Educator maintains a journal of transactions per each Activity Type that the Educator offers.

All the Activity Types are grouped into courses that are further grouped into larger entities such as subjects and areas of knowledge. This grouping can be stored as the Activity Type Graph G_A with the following properties:

1° G_A is a directed graph:

$$G_A : \langle V : \{\text{Vert}\},\ e_{out} : \text{Vert} \to \{\text{Vert}\} \mid \text{rest}\rangle \tag{1}$$

2° Each vertex of G_A is associated with depth:

$$G_A : \langle d : \text{Vert} \to \text{Int} \mid \text{rest}\rangle \tag{2}$$

3° Law of pointing down:

$$G_A : \langle v \in e_{out}(u) \implies d(v) > d(u)\rangle \tag{3}$$

4° G_A has special *et cetera* vertices u:

$$\forall\ v \in V\ \exists u\ (u \in e_{out}(v) \wedge e_{out}(u) = \emptyset) \tag{4}$$

The example of the Activity Type Graph (ATG) is shown in Figure 2. The vertex v of the graph is a *leaf* if $e_{out}(v) = \emptyset$. Otherwise we call it an *internal vertex*. Every internal vertex of the graph has a special *etc.* child (some of these are ommitted in the figure).

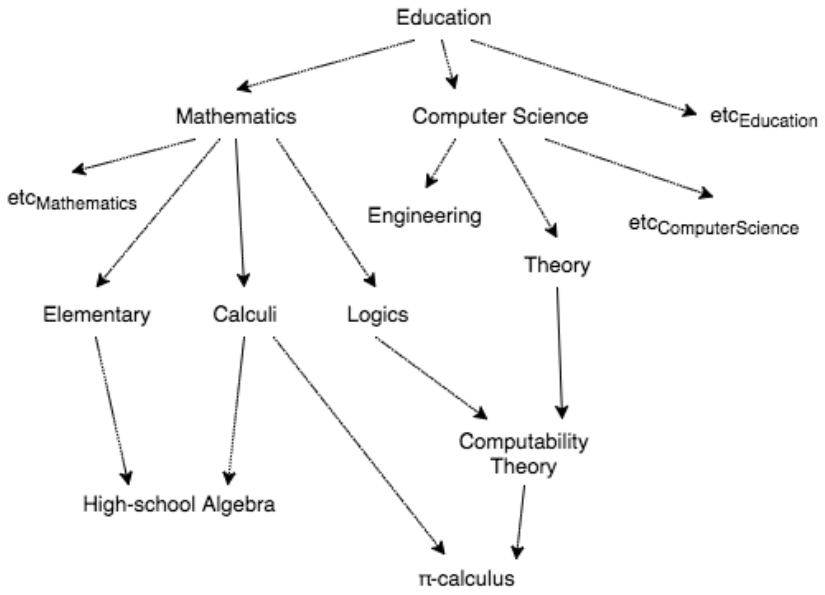

Figure 2: An example of the Activity Type Graph. Some of the vertices are not shown

The need for *etc.* vertices arises from the fact that not all of the Educators teach courses exactly in leaves — some of them offer general courses that provide just the necessary background. For example, some of the universities teach the basic "Computer science" course, that contains the basics of the discipline. In this case, when the particular category is hard to define, the university would use the $\text{etc}_{\text{ComputerScience}}$ vertex.

On the protocol level, the Educators can announce that they teach a particular course, but can not modify the Activity Type Graph structure. The structure of the graph is maintained by the core developers and updated upon request from the Educators.

For every pair of vertices (v, u), $weight(v, u)$ defines how the score of a course from the field of study u affects the summary grade for the field of study v. Let's define $weight(v, u) = \frac{1}{1+d(u)-d(v)}$ if u reachable from v, and $weight(v, u) = 0$ otherwise. After that we can define $avgGrades_{subjectId}$ as a weighted average with weights described above.

3.3 Search queries

An educator can answer one of the following queries:

- For a set of pairs $(subjectId_1, minGrade_1)$, $(subjectId_2, minGrade_2)$, ..., $(subject_n, minGrade_n)$ and some $Count$, find no more than $Count$ students with grades satisfying the following inequalities:

$$\begin{cases} avgGrade_{subjectId_1} >= minGrade_1 \\ avgGrade_{subjectId_2} >= minGrade_2 \\ \vdots \\ avgGrade_{subjectId_n} >= minGrade_n \end{cases}$$

- For the given identifier of a student, return all info about this student.
- For given time range, activity and student, return hashes of work (merkle tree root) of assignments submitted. Returning a root will allow the opposite side of the trade to prove that transmitted data was corrupted.

3.4 Private chain

Every educator has a private chain. It stores the data about students, and can generate answers for the queries described above.

Private chain comprises of two main data structures:

- Set of transactions batched into blocks. Every block contains a list of transactions packed into a Merkle tree.
- Links to the transactions stored in the B+-tree with keys (studentId, studentGrade). Indexes constructed in such a way that more popular activities go first.

The structure of the private block is shown in Figure 3. The block consists of a public *header* that the Educators relay to the Witnesses, and the private *body* that remains in the educational institute until it receives a data disclosure request.

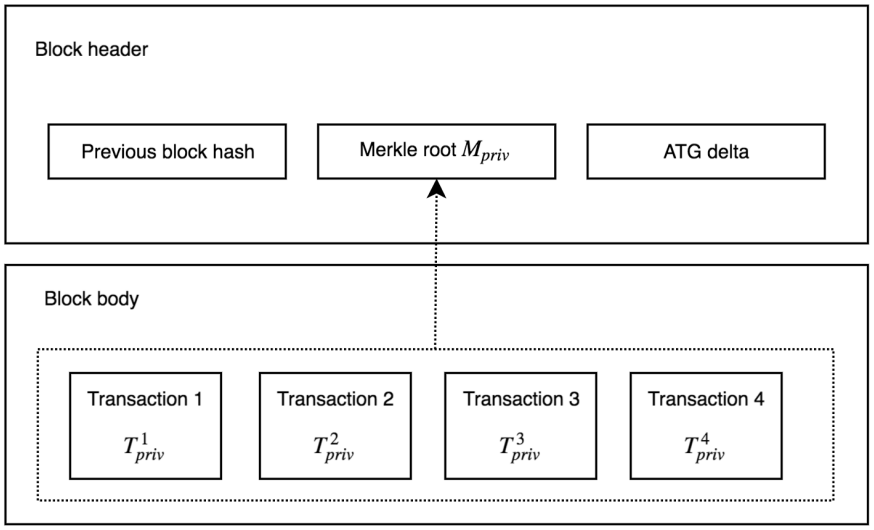

Figure 3: Private block structure

During the educational process the Educators emit atomic *private transactions*. These tansactions represent the modifications to the journal of academic achievements (thus, making a transaction means appending the data to the journal). The transactions can be of the following types:

- student enrolls in a course;
- student gets an assignment;

- student submits an assignment;
- student gets a grade for an assignment;
- student gets a final grade for the course.

The first two types should be intiated by a student, and should include student's signature to prevent spam from partially-honest educator. The structure of the transaction is shown in Figure 4.

Let us denote an i-th transaction in a block as T^i_{priv}. The Educators group the transactions that occured during the current block time slot, and construct a Merkle tree [**?**] for these journal modifications:

$$M_{priv} = \texttt{mtree}(T^i_{priv}) \tag{5}$$

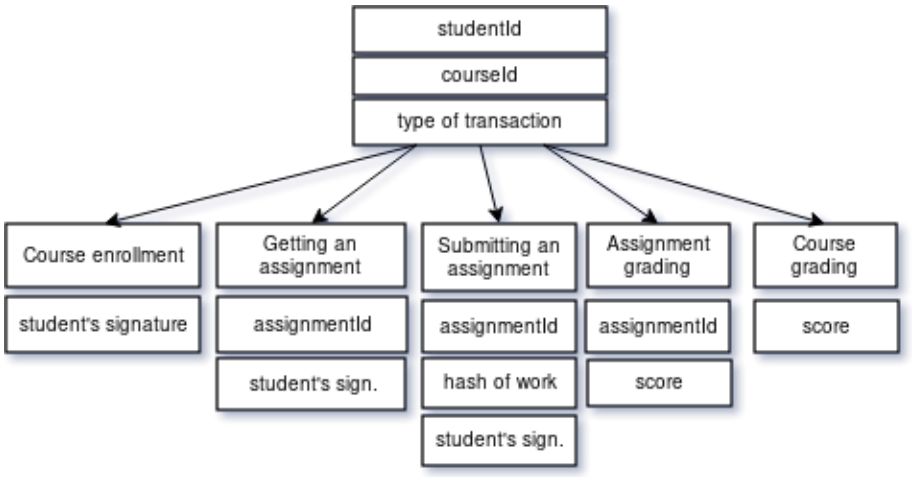

Figure 4: Transaction structure

The Educator's private block body comprises an ordered set of Merkle-authenticated transactions. These transactions are indexed so that the Educator can quickly find a particular transaction that satisfies some predicate.

The private block header consists of the transactions Merkle root along with the previous block hash and the information on the Activity Type Graph modifications (ATG delta). The *ATG delta* part allows the Educators to inform the Witnesses and the Archivists of the modifications to the courses they teach.

An Educator collects private transactions into the blocks with no more than K transactions per each block. After that, an Educator submits signed block header to the Witnesses so that private transactions can be confirmed by the public chain. Thus, the private blocks form a publicly verifiable chain of events.

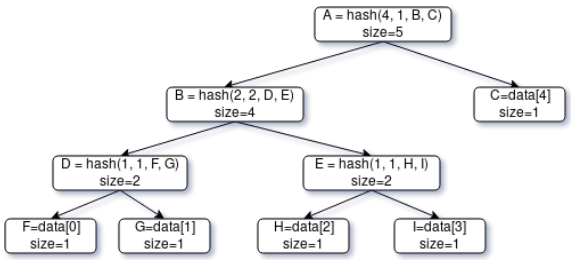

Figure 5: Example of Merkle tree

To incentivize Witnesses to include private block headers into the public chain, an Educator should pay some amount of coins per each private block. We should take into consideration that an educator may be both a local tutor and some big university. Depending on that, a number of transactions per each block, as well as paying capacity, may differ. So the cost of a digest publication linearly grows with the size. To achieve the ability to prove the size of the Merkle tree, we will store it together with a hash in each node(as shown in 5). So every transaction disclosure will also verify the size. [The alternative idea described in the appendix].

3.5 Public chain

The Witnesses maintain a public chain – a distributed ledger that contains publicly available information. If one wishes to perform a transaction on the public chain, she has to pay a certain fee that serves two purposes. First of all, the fee incentivizes the Witnesses to participate in the network and issue new blocks. Second, by requiring a fee for each transaction, we protect the public ledger from being spammed.

We present the structure of the public blocks in Figure 6. The public ledger contains the following information:

1. Modification history of the Activity Type Graph.
2. Private transaction proofs.
3. Account balances and value transfer history.

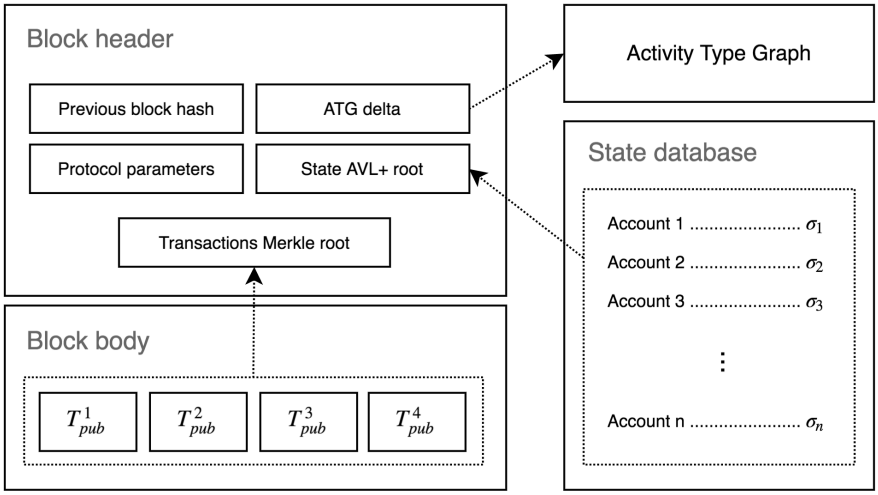

Figure 6: Public block structure

There are two major ways to store the account balances and other state information: UTXO and account-based architectures. UTXO is an unspent transaction output, that contains some predicate – a condition that has to be fulfilled in order to spend the coins. To prove the money ownership, the spender provides a witness – an input that makes the predicate true. Thus, the UTXO-based architecture requires the transactions to be stateless, effectively limiting the application domain [**?**]. The unspent outputs with an associated state can be treated as smart-contracts in the account-based architectures like Ethereum [**?**]. The state is stored in an off-chain storage – the state database. The transactions are treated as the modifications of the world state.

Disciplina uses an account-based ledger with contracts programmable in Plutus language [**?**]. Each account has an associated state, which comprises the account balance and other information (e. g. $\log L$ of a data disclosure contract). The world state is a mapping between accounts and their states. In order to make this mapping easily verifiable, we use a structure called the *authenticated AVL+ tree* introduced in [**?**]. This structure is based on state-of-the-art research that enables for faster verification of the mapping and allows us to never disclose it: the Witnesses would not have to store the whole blockchain like Bitcoin or Ethereum nodes do. Rather, they would just have to check the private block headers in order to confirm that none of the private blocks were tampered with.

The recent achievements in the field of consensus protocols, like the provably secure Ouroboros [**?**], allow us to build a public chain based on the Proof of Stake consensus rules. Thus, we can increase the transaction speed and drop the need for the expensive mining. However, with mining being dropped, we need to provide incentives for the Witnesses to maintain the chain and participate in the network.

3.6 Fair CV

One of the main goals of the Disciplina platform is to provide a way for the Students to easily prove their educational records. We propose to duplicate the records in the Student's *digital CV*. This CV contains all the records that the parties have generated during the Student's educational process along with the validity proofs of that data (see Figure 7).

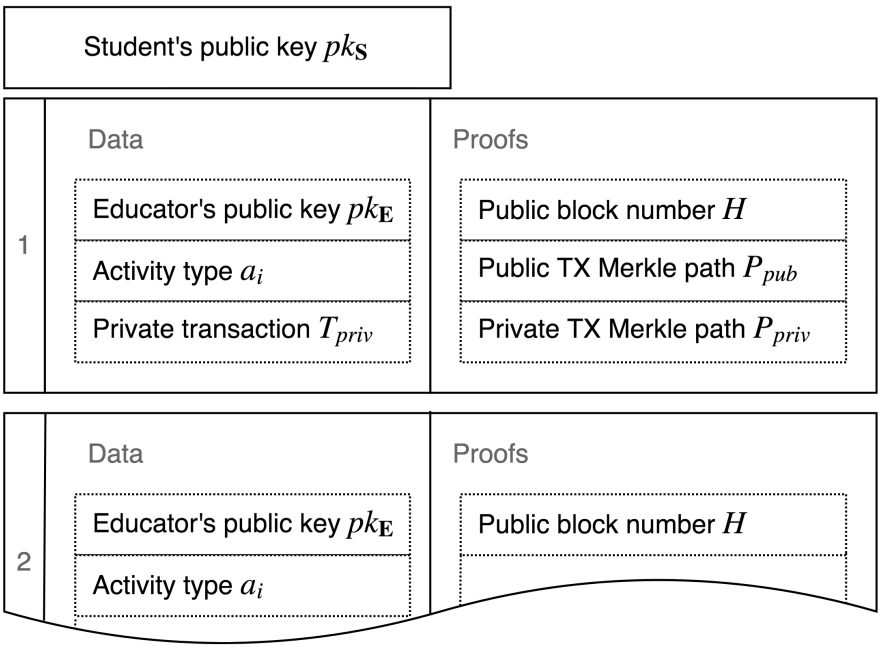

Figure 7: Student's authenticated CV

In order to prove that some transaction actually occurred in some private block of the concrete Educator, the student has to provide the cryptographic proofs along with the actual data. The cryptographic proof of the inclusion of an element in an authenticated data structure is generally a path of hashes. Let us denote the path of the element e in some authenticated data structure X as $\texttt{path}(e, X)$. Thus, the Student has to provide the following data:

- The Student's and the Educator's public keys $pk_{\mathbf{S}}$ and $pk_{\mathbf{E}}$.
- The course a_i and the a private transaction T_{priv} with the score.
- The Merkle path of the transaction in the journal: $P_{priv} = \texttt{path}(T_{priv}, M_{priv})$, where M_{priv} is a Merkle tree of the transactions in the private block.
- The public block number H and the Merkle path of the transaction T_{pub} that pushed the private block into the public chain: $P_{pub} = \texttt{path}(T_{pub}, M_{pub})$, where M_{pub} is a Merkle tree of the transactions in the block H.

Having this data one can prove the occurrence of a certain transaction in one of the Educator's private blocks without the need to request any data from the Educator during the validation process. Thus, any party can check the validity of the Student's CV for free if the Student wishes to disclose it.

Let $\rho(e, P)$ be the function that substitutes the element e in path P and computes the root hash of the authenticated data structure. Then the validation process is as follows:

1. Query the public chain to find the block H and obtain the Merkle root of the transactions: $\texttt{root}(M_{pub})$.
2. Check whether $\rho(T_{pub}, P_{pub}) = \texttt{root}(M_{pub})$.
3. Check that the public transaction T_{pub} was signed with the Educator's public key $pk_{\mathbf{E}}$.

4. From the public transaction T_{pub} obtain the Merkle root of the private transactions: $\texttt{root}(M_{priv})$.

5. Check that $\rho(T_{priv}, P_{priv}) = \texttt{root}(M_{priv})$.

These validation steps can prove that an Educator with a public key $pk_{\mathbf{E}}$ issued a transaction T_{priv} in one of its private blocks. One can attribute the $pk_{\mathbf{E}}$ to a particular real-world educational institution by checking the Educator's certificate as described in Section 3.1.

3.7 Data Disclosure

The process of data disclosure involves direct communication between a particular Educator, willing to disclose a part of the data, and an interested party **B** (e. g. a recruiter), willing to pay for this data. Suppose an Educator **E** has some data D. To mitigate the risk of secondary market creation, one should ensure that the majority of the data remains in the private blocks. We propose to increase the cost of the data exponentially to its size, thus incentivizing the buyers to make as accurate requests as possible.

Before the deal **E** ought to perform some preparation steps. **E** should:

1. Divide D into N chunks of size no more than 1 KiB:

$$D = \bigoplus_{i=1}^{N} d_i, \quad \texttt{sizeof}(d_i) \leq 1 \text{ KiB} \tag{6}$$

2. Compute a Merkle root of the unencrypted chunks:

$$R_0 = \texttt{root}(\texttt{mtree}(D)) \tag{7}$$

3. Generate a symmetric key k

4. Encrypt each d_i with k and make an array of encrypted chunks:

$$D = \{\texttt{E}_k(d_1),\ \texttt{E}_k(d_2),\ ...,\ \texttt{E}_k(d_N)\} \tag{8}$$

5. Compute a Merkle root of the encrypted chunks:

$$R = \texttt{root}(\texttt{mtree}(D)) \tag{9}$$

6. Determine the size of the data she is going to reveal:

$$s = \texttt{sizeof}(D) \tag{10}$$

7. Calculate the cost of the data (α is some constant coefficient):

$$C_D = \alpha \exp(s) \tag{11}$$

The protocol fairness is guaranteed by a contract on the public chain. The contract is able to hold money and is stateful: it is capable of storing a log L with data. All the data that parties send to the contract are appended to L.

1. **E** creates a contract on the public chain and sends the following data to the contract: $\texttt{Sig}_{\mathbf{E}}(C_D,\ \texttt{sizeof}(D), R, R_0)$. She also sends a predefined amount of money C_E to the contract address. C_E is a security deposit: if **E** tries to cheat, she would lose this money.

2. The buyer generates a new keypair ($pk_{\mathbf{B}},\ sk_{\mathbf{B}}$), and sends C_D worth of money to the contract address. Along with the money, **B** sends the public key $pk_{\mathbf{B}}$ of the newly generated keypair, and the size of the data.

3. **E** transfers the encrypted data chunks D to the buyer. **B** computes the Merkle root R' and the size s' of the received data D':

$$R' = \texttt{root}(\texttt{mtree}(D')) \tag{12}$$

$$s' = \texttt{sizeof}(D') \tag{13}$$

4. **B** makes a transaction with a receipt $\texttt{Sig}_{\mathbf{B}}(\{R',\ s'\})$ to the contract address. The parties can proceed if and only if the following is true:

$$(R' = R)\ \wedge\ (s' = s) \tag{14}$$

   Otherwise, the protocol halts.

5. **E** sends $\texttt{Sig}_{\mathbf{E}}(\texttt{E}_{\mathbf{B}}(k))$ to the contract.

6. **B** decyphers and checks the received data. In case some data chunk $e_i \in D$ is invalid, **B** sends a transaction with $\{sk_{\mathbf{B}},\ e_i,\ \texttt{path}(e_i,\ \texttt{mtree}(D))\}$ to the contract. By doing so, **B** reveals the data chunk d_i corresponding to the encrypted chunk e_i. She also shares proof that e_i was indeed part of a Mekle tree with root R. The contract checks the validity of d_i and decides whether **B** has rightfully accused **E** of cheating.

The on-chain communications of the parties (steps 2, 4, 5, 6) are bounded by a time frame τ. In order for the transaction to be valid, the time Δt passed since the previous on-chain step has to be less than or equal to τ. In case $\Delta t > \tau$ the communication between the parties is considered over, and one of the protocol exit points is automatically triggered. The protocol exit points are described in detail in Table 1.

Table 1: Data disclosure protocol exit points

Condition	**Step**	**Consequence**
$\Delta t > \tau$	2	**B**, **E** get their money back because **E** wasn't able to correctly transfer the data to **B**.
$\Delta t > \tau$	4	
$R' \neq R$	4	
$s' \neq s$	4	
$\Delta t > \tau$	5	**B**, **E** get their money back because **B** has received the encrypted data, but **E** nas not been able to share the key k for it
$\Delta t > \tau$	6	**E** gets C_E and C_D: **E** correctly shared data to **B**
Protocol finishes		The dispute situation. In case **B** proofs **E** cheated, **E** loses her security deposit C_E. Otherwise, **E** receives both C_E and C_D.

3.8 Smart-contract implementation

3.8.1 Accounts

There are 2 kinds of accounts:

- Personal: created for each client, directly belongs to that client;
- Robot: created by client, doesn't belong directly to that client (and anyone else); represents smart contract.

Robot account should contain, aside from token mapping:

- Data storage for smart contract state;
- The code to control the account, compiled to Plutus Core language.

The Plutus Core language allows declaring a module with exported (public) methods. These methods will be the public API for the account.

One can evaluate the Plutus Core code itself (not just call public methods) if account directly belongs to her. Personal accounts don't have any persistent associated code to control them.

3.8.2 Scripting

The Plutus language [**?**] will be used to program Robot nodes. Any interaction with account is done via Plutus code.

The evaluation cost is collected and summed into Fee.

"Sandbox run" can be performed, to check costs and programming errors.

We will call natively-implemented functions to be called from Plutus Core "NIFs".

3.8.3 NIFs

NIF is an (ortogonal to each other) action to be invoked on Account. NIF is the only source of changes in Account.

Any operation on the Account to be implemented should be represented as a call to NIF OR as a sequence of NIF-calls. This will allow us to reason about security/validity of operations with more confidence and limit the access and scope of each operation.

Each NIF has an invocation cost.

3.8.4 Transaction

We will cover "simple money transfer" and "data disclosure" transactions in this section.

"Simple money transfer" transactions will be implemented as transactions with empty "function call" field.

Transferral transaction must contain:

- Sender signature;
- Receiver signature;
- Amount and kind of funds transferred (must be non-zero overall);
- "Nonce" to prevent double-invocation.
- (Optional) textual message for the receiver;
- (Optional) function call to be invoked upon transaction approval.
- Digest of all other fields, encrypted with Sender private key.

Transaction is the only way for accounts to interact.

Function call (if present) will contain the name of exported function and arguments to be supplied upon invocation.

The function would be invoked like that:

```
function-name(Seller-signature, amount, value1, value2, ..., valueN)
```

On successful code invocation, money will be transmitted to the Target account and the costs will be demanded from the Sender. If code fails, the transaction is not published as successful and is rejected.

If there is not enough money supplied for the operation or the code raised an error, whole transaction will fail.

If there is no function call in transaction, the code invocation is assumed successful.

The fee for transaction approval will depend on NIFs invoked and will have its cost calculated as `K * (NifInvoked + C * sqr(NifInvoked))` where

- `K` is a global tuning parameter;
- `C` is a parameter to cut off long transactions. It is small enough, so fast-ending transactions will not notice it; in the same time big transactions (like the one trying to evaluate `akkermann(10, 10)`) will run out of gas.

3.8.5 Smart-contract mechanics

We assume that we have 2 sides:

- Buyer
- Seller.

"Gas" below is the estimation of the operation cost. The name and idea is taken from Etherium [?].

Smart contracts would work as follows. Buyer invokes a transaction which runs code directly on his account, that constructs a robot with following exported methods

- `initiate();`
- `accept-fee();`
- `accept();`
- `reject(block, proof);`
- `refuse();`
- `check-time(),`

carrying `Sum + Gas` amount of currency and some `predicate` to check the data.

Here is the state machine of that Robot-account:

```
(0) Account was created.
  "initiate" from Buyer:
    Send an invitation to buyer
    AND GOTO 1

  "reject" from Buyer: cleanup if something is wrong
    GOTO 4

(1) The trade was started.
      "accept" from Buyer
  AND "accept" from Seller: both accepted, terminating contract
    GOTO 3

  "reject" from Buyer
    GOTO 2

  "refuse" from Seller: in case smart contract doesn't hold enough currency.
    GOTO 4

  "check-time" from Seller
    when time-spent > TIMEOUT
      GOTO 3

  "check-time" from Buyer
    when time-spent > TIMEOUT
    AND  Seller did not respond in time
      GOTO 4

(2) Arbitration.
  The robot invokes 'check-block-and-proof' function.
  If it signals that Buyer is right (block invalidates the proof OR the predicate fails),
    GOTO 4
  else if Seller is right
```

```
    GOTO 3

(3)
  Sum is sent to Seller.
  GOTO 4.

(4) Cleanup. The account is closed and all remaining money
   are sent to the Buyer.
```

3.8.6 Example

Lets assume there are:

- Seller which has declared that he has his students' Linear Algebra marks for Nov, 2018 worth 500 tokens (signed in some private Merkle tree);
- Buyer which has 600 tokens available.

We will consider three cases:

- Seller tried to send Buyer garbage instead of data;
- Buyer tried to blame Seller in giving her invalid data with data being completely valid (in terms of signature and predicate).
- Buyer decided to not `accept()` the contract.

All trades will have same initial part, so we will branch when nessessary. The trades will go as follows:

- Buyer formulates a predicate to check that data corresponds its description.
- She estimates the gas cost to be covered by 20 tokens max.
- Then she creates smart account using the scheme above with 500 + 20 tokens and the predicate.
- She checks that everything is right and invokes `initiate()`, which notifies Seller.
- Seller accepts and sends Buyer encrypted data via prvate channel, along with key.

    1. Seller tries to send garbage:
        - Buyer decrypts the data and finds that one block signature is invalid.
        - Then she invokes `reject(unencrypted-block, proof)` to start arbitration.
        - The smart account performs `check-block-and-proof(block, proof)` and finds that Buyer was right.
        - Then it returns all remaining funds to Buyer.
    2. Buyer tries to blame Seller with valid data:
        - Buyer selects the block to call "invalid".
        - Then she invokes `reject(unencrypted-block, proof)` to start arbitration.
        - The smart account performs `check-block-and-proof(block, proof)` and finds that Buyer was not right.
        - Then it send 500 tokens to Seller and all remaining currency to Buyer.
    3. Buyer receives the data, but remains silent:
        - Seller after some time calls `check-time()`.
        - If the timeout is expired, 500 tokens are sent to Seller.
        - All remaining currency is sent to Buyer.

4 Future work

The current architecture of the Disciplina platform heavily relies on the external entities – the Archivists – to add a layer of trust and provide ratings for the educational institutions and private teachers. However, it is possible that such external entities would provide unfair ratings: for example, they could ignore the existence of private teachers, thus making their contributions less valuable. The reliance on the third parties can be avoided if we carefully integrate the algorithm of rating computation into our architecture. The ratings would be based on the on-chain sources of information and provide equal opportunities for both private teachers and large educational institutions. However, integrating the rating system into the arhitecture poses several design challenges that we have to solve.

5 Conclusion

In this paper we presented the architecture of the Disciplina platform. The described architecture provides a way to store educational records in the blockchain while preserving the privacy of these records. The concepts of private chains and a digital CV make it possible to verify the educational records of a particular person. The Archivist entities provide reputational semantics to the educational institutes listed in the digital CVs.

We developed our platform not only as the source of trust, but also as a database of the students from all over the world. We believe that the data that is stored in the system has a value in itself. The need to disclose this data was also addressed in the paper: we described a mechanism for the fair data trade and the measures against the secondary market creation.

A Notations

Notation	Description
$\mathbf{A}$	A party that takes part in the protocol
$\mathtt{H}(m)$	Result of applying a collision-resistant hash-function $\mathtt{H}$ to a message m
$\mathtt{mtree}(a)$	Merkle tree of the data array a
$\mathtt{root}(M)$	Root element of the Merkle tree M
$\mathtt{path}(e, M)$	Path of the element e in the Merkle tree M
k	Symmetric key
$pk_{\mathbf{A}}$, $sk_{\mathbf{A}}$	Public and secret keys of $\mathbf{A}$
$\mathtt{E}_k(m)$	Symmetric encryption with the key k
$\mathtt{E}_{\mathbf{A}}(m)$	Asymmetric encryption with the key $pk_{\mathbf{A}}$[1]
$\mathtt{Sig}_{\mathbf{A}}(m)$	Tuple $(\mathbf{A}, m, sig(sk_{\mathbf{A}}, H(m)))$, where sig is a digital signature algorithm[1]
$\mathtt{sizeof}(m)$	Size of m in bytes
$\bigoplus$	Binary string concatenation

A Data Object Storage

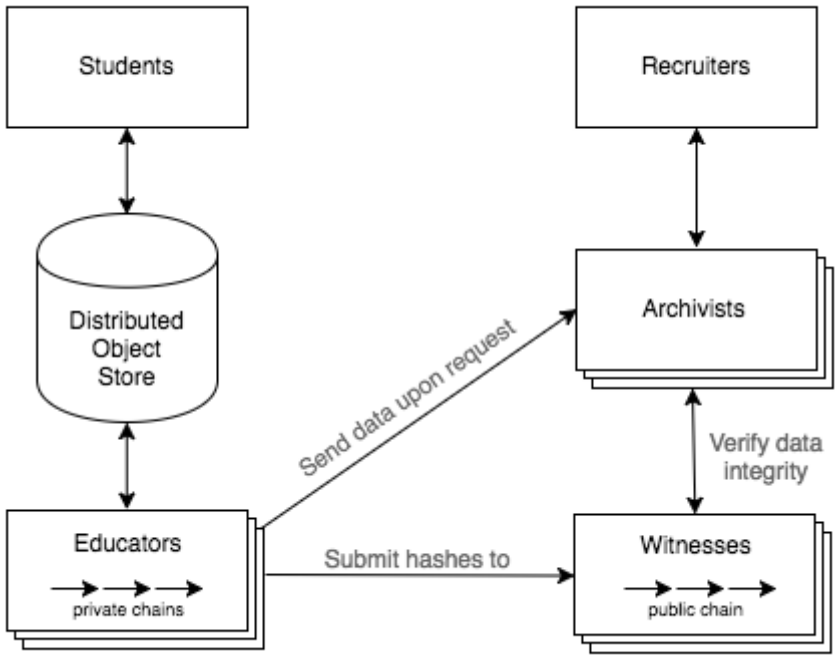

Figure 8: Key entities of the Disciplina platform

A Partially centralized educators

In the Section 3.4 we have concluded that the cost of the private hash publication should depend on the size of the Merkle tree. Our alternative idea how to solve inequality between local tutors and big universities is to delegate block generation to the special platforms for tutors(e.g. teachmeplease.com).

[1]The particular keys $pk_{\mathbf{A}}$ and $sk_{\mathbf{A}}$ belonging to the party $\mathbf{A}$ are generally deducible from the context