Disciplina: Blockchain for Education

Kirill Kuvshinov[1], Ilya Nikiforov[2], Jonn Mostovoy[3], Dmitry Mukhutdinov[4], Kirill Andreev[5], and Vladislav Podtelkin[6]

[1, 2]Teach Me Please, https://teachmeplease.com
[3, 4, 5, 6]Serokell, https://serokell.io

Version 0.6.1
January 30, 2018

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 [9] blockchains had been primarily used as a mechanism for value transfers. With the growth of the Ethereum platform [13], 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 [6], Hyperledger [2], Kadena [4] 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, [12] and [3] 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 [11]. 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 1.

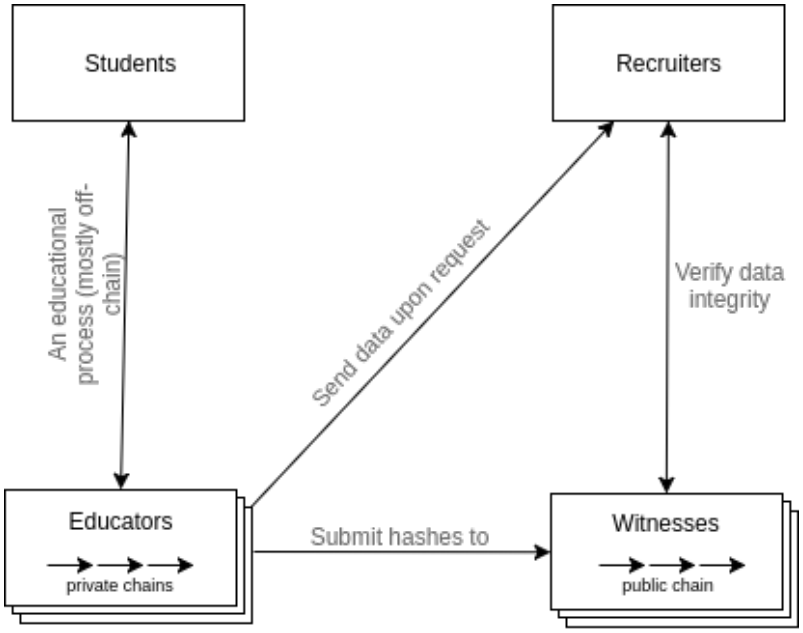

Figure 1: Key entities of the Disciplina platform

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 with a score in range [0..100], 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.

The Recruiters are the entities interested in gathering data about students from educational institutions. They buy this data from Educators using a secure data disclosure protocol, described in detail in section 3.7. Validity and security of every data trade is also ensured by Witnesses, because corresponding transactions and actions of each party are also stored in public blockchain.

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 disallowing a new Educator to join the network without an approval from another Educator. Educators are supposed to rate another Educators basing on off-chain sources of information – such as a publication on an official site of a university, which claims that given Disciplina public key is issued by this university. By approving each other, Educators form a *web of trust*. Ratings of Educators are backed up by ratings of Educators which trust them.

3.2 Activity Type Graph

When a Recruiter makes a request to one of the Educators, the Educator 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).

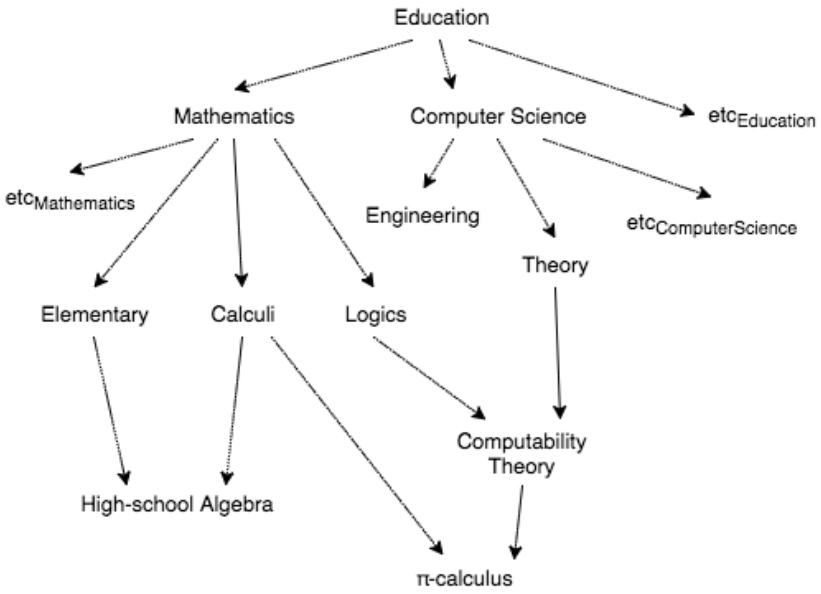

Figure 2: An example of the Activity Type Graph. Some of the vertices are not shown

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 $weight(v, u) = (d(u) - d(v) + 1)^{-1}$ if u reachable from v, and $weight(v, u) = 0$ otherwise. The motivation of the aforementioned weights is that less specific subject implies the wider knowledge. 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 assignment hash, return the document itself.

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.

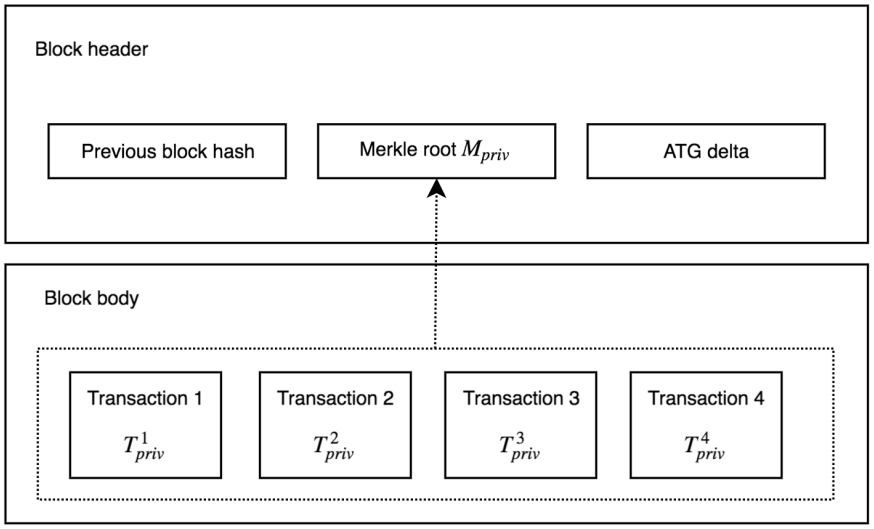

Figure 3: Private block structure

During the educational process the Educators emit atomic *private transactions*. These transactions 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 [8] for these journal modifications:

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

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.

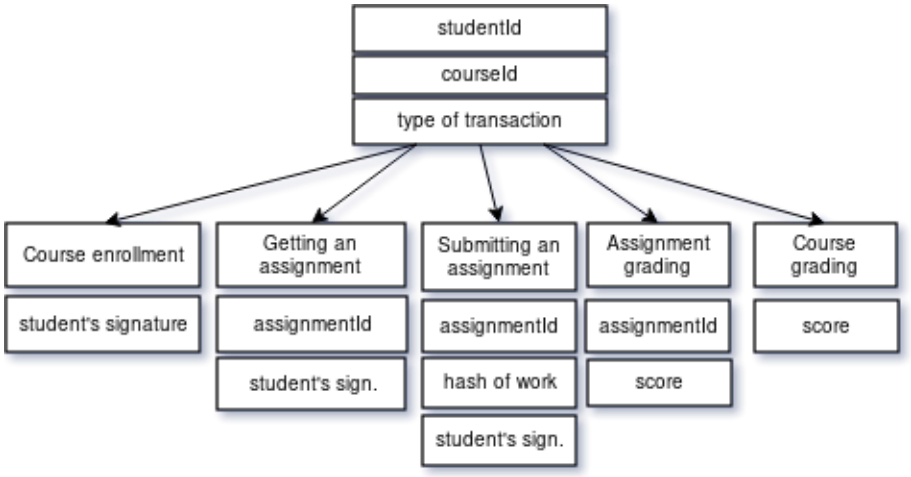

Figure 4: Transaction structure

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 of the modifications to the courses they teach.

An Educator collects private transactions into the blocks with no more than K_{max} 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.

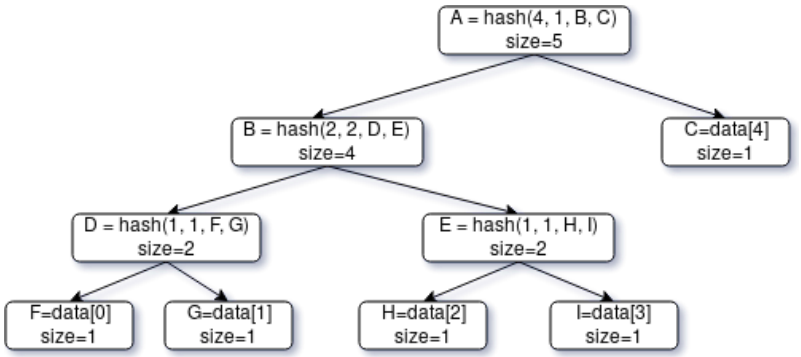

Figure 5: Example of sized Merkle tree

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 should linearly grow with the size of a block. Let the cost for publishing a public block header be

$$C_{pub}(B) = \alpha_{pub} + \beta_{pub} \cdot N_{tr}(B) \tag{6}$$

, where $N_{tr}(B)$ is the number of transactions in private block B and α_{pub} and β_{pub} are parameters of the network – a small constant fee and a linear price coefficient accordingly.

To achieve the ability to prove the number of transactions in the Merkle tree, we will store it together with a hash in each node (as shown in Figure 5). The proof will only be valid if the Educator fills in the sizes correcctly, so there is no incentive for Educators to lie about the size of the tree.

We also consider a possibility for small educators to form pools and release blocks together in order to reduce costs for each individual educator. See appendix A.2 for details.

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 block headers.
3. Account balances and value transfer history.

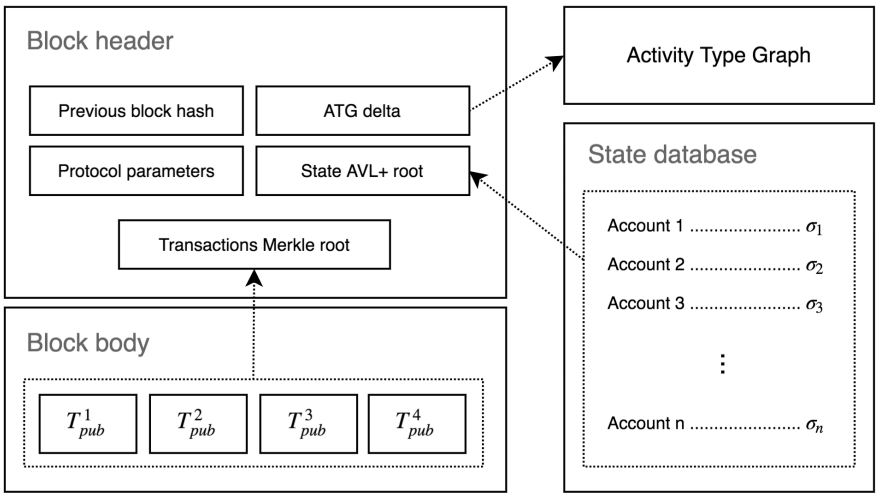

Figure 6: Public block structure

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 [1]. The unspent outputs with an associated state can be treated as smart-contracts in the account-based architectures like Ethereum [13]. 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 [7]. 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 [10].

The recent achievements in the field of consensus protocols, like the provably secure Ouroboros [5], 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.

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).

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

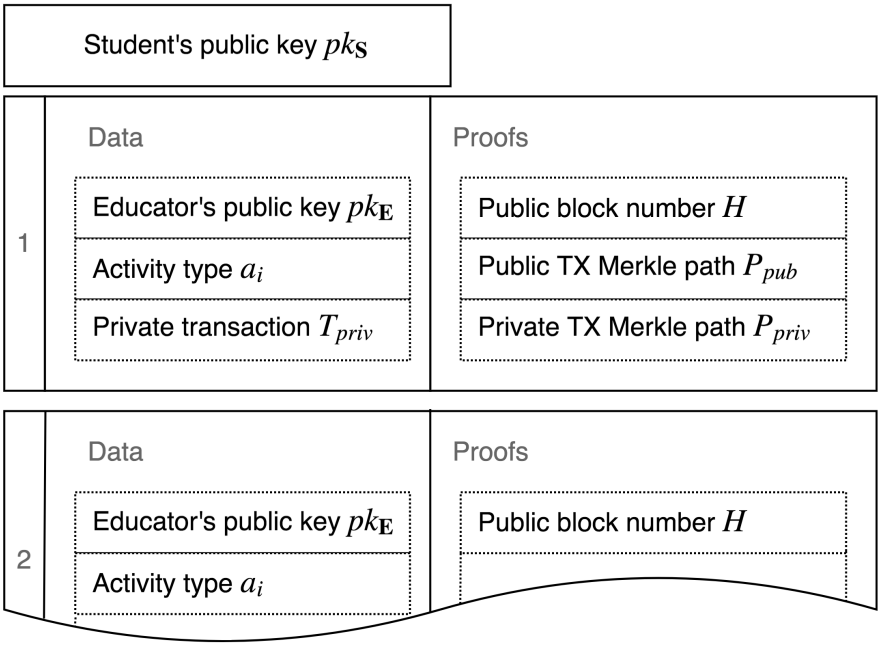

Figure 7: Student's authenticated CV

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

Disciplina architecture supports two types of data disclosure requests:

1. Request for a set of authenticated private transactions satisfying some predicate (see details in Section 3.3)
2. Request for object disclosure

Here we describe a protocol of fair data trade between the Educator as a seller and some interested party as a buyer. Despite a few variations the protocol is almost the same for all three types of the data disclosure requests. We first lay out the private transactions disclosure protocol. Then we describe modifications to the protocol so that one can apply it to other types of data.

The process of data disclosure involves direct communication between a particular Educator **E**, willing to disclose a part of the data, and an interested party **B** (e. g. a recruiter), willing to pay for this data. Suppose **E** has some data D. In case of private transactions D is a set of *authenticated transactions*, i. e. tuples $(T_{priv}, P_{priv}, H, P_{pub})$. As shown in Section 3.6 this data along with the educator's public key is enough to prove that a certain transaction T_{priv} actually occurred in some private block of the given educator.

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 is appended to L.

1. The buyer **B** sends a signed search query $\texttt{Sig}_{\mathbf{B}}(Q)$ directly to the seller **E**.
2. Let D be a set of authenticated transactions relevant for the query Q. **E** divides D into N chunks. When disclosing private transaction, one chunk d_i is a transaction with proofs that it was included in a certain private block:
$$d_i : (T^i_{priv}, P^i_{priv}, H^{(i)}, P^i_{pub}) \tag{7}$$
3. **E** generates a symmetric key k and encrypts each d_i with k. Then she makes an array of encrypted chunks:
$$D_{\blacksquare} = \{\texttt{E}_k(d_1),\ \texttt{E}_k(d_2),\ ...,\ \texttt{E}_k(d_N)\} \tag{8}$$
4. **E** computes the size of the encrypted answer $s = \texttt{sizeof}(D_{\blacksquare})$, the cost of this data $C_D \sim s$, and the Merkle root of the data $R = \texttt{root}(\texttt{mtree}(D_{\blacksquare}))$.
5. **E** sends $\texttt{Sig}_{\mathbf{E}}(C_D,\ s,\ R,\ \texttt{H}(Q))$ directly to the buyer.
6. If buyer agrees to pay the price, she generates a new keypair $(pk_{\mathbf{B}},\ sk_{\mathbf{B}})$. Then she initializes the contract with the data provided by the Seller, search query Q, its own temporary trade public key $pk_{\mathbf{B}}$ and C_D amount of money.
7. If **E** agrees to proceed, she 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.
8. **E** transfers the encrypted data chunks $D_{\blacksquare}$ directly to the buyer. **B** computes the Merkle root R' and the size s' of the received data $D_{\blacksquare}{}'$:
$$R' = \texttt{root}(\texttt{mtree}(D_{\blacksquare}{}')) \tag{9}$$
$$s' = \texttt{sizeof}(D_{\blacksquare}{}') \tag{10}$$
9. **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:
$$(R' = R)\ \wedge\ (s' = s) \tag{11}$$
Otherwise, the protocol halts.
10. **E** sends $\texttt{Sig}_{\mathbf{E}}(\texttt{E}_{\mathbf{B}}(k))$ to the contract.
11. **B** decyphers and checks the received data.

- In case all the data is correct the Buyer sends a signed accept to the contract.
- In case some data chunk $e_i \in D_{\blacksquare}$ is invalid, **B** sends

$$\texttt{Sig}_{\mathbf{B}}(\{\ sk_{\mathbf{B}},\ e_i,\ \texttt{path}(e_i,\ \texttt{mtree}(D_{\blacksquare}))\ \})$$

  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 Merkle tree with root R. The contract checks the validity of d_i and decides whether **B** has rightfully accused **E** of cheating.
- In case chunks d_i and d_j have duplicate entries, **B** sends

$$\texttt{Sig}_{\mathbf{B}}(\{\ sk_{\mathbf{B}},\ e_i,\ \texttt{path}(e_i,\ \texttt{mtree}(D_{\blacksquare})),\ e_j,\ \texttt{path}(e_j,\ \texttt{mtree}(D_{\blacksquare}))\ \})$$

  to the contract. The contract checks whether d_i and d_j do indeed have duplicate entries and blames **E** for cheating if it is true.

The contract considers the data chunk d_i valid if and only if:

1. The transaction in d_i is unique.
2. The transaction in d_i has valid proofs of existence (like described in Section 3.6).
3. The transaction in d_i make the predicate Q true.

The on-chain communications of the parties (steps 7, 9, 10, 11) 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**
$\Delta t > \tau$	7	**B**, **E** get their money back because **E** wasn't able to correctly transfer the data to **B**.
$\Delta t > \tau$	9	
$R' \neq R$	9	
$s' \neq s$	9	
$\Delta t > \tau$	10	**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
$\Delta t > \tau$	11	**E** gets C_E and C_D: **E** correctly shared data to **B**
`accept` from **B**	11	
`reject` from **B**	11	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.

The proposed algorithm (though with some modifications) can be applied to object disclosure requests. Here we define these modifications:

- Q : $\texttt{root}(\texttt{mtree}(\textit{Object}))$ – query by the object hash.
- d_i : $(\textit{chunk},\ \texttt{path}(\textit{chunk},\ \texttt{mtree}(\textit{Object}))$ – the data being revealed is an *object:* uncategorized blob of data relevant to a particular transaction. The object is split into chunks of size no more than 1 KiB and transferred along with proofs.
- Validation: check that a chunk is indeed a part of the object with root Q.

4 Future work

The current architecture of the Disciplina platform heavily relies on the fact that a new Educator should gain acceptance from other Educator to join the network, and ratings of a new Educator are determined by other Educators accepting it. However, it is possible that other Educators would provide unfair ratings: for example, they could ignore the existence of private teachers, thus making their contributions less valuable, or purposefully lower the ratings of competitors entering the network.

Such problems 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. Educational institutions are connected in a web of trust to provide credibility for each institution and, consequently, to digital CVs of their alumni.

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.

References

[1] Iddo Bentov, Ranjit Kumaresan, and Andrew Miller. "Instantaneous Decentralized Poker". In: *arXiv preprint arXiv:1701.06726* (2017).

[2] Christian Cachin. "Architecture of the Hyperledger blockchain fabric". In: *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*. 2016.

[3] Peter Devine. "Blockchain learning: can crypto-currency methods be appropriated to enhance online learning?" In: (2015).

[4] *Kadena: Scalable blockchain*. https://kadena.io/.

[5] Aggelos Kiayias et al. "Ouroboros: A provably secure proof-of-stake blockchain protocol". In: *Annual International Cryptology Conference*. Springer. 2017, pp. 357–388.

[6] Jae Kwon. "Tendermint: Consensus without mining". In: *Draft v. 0.6, fall* (2014).

[7] Darryl McAdams. *Formal Specification of the Plutus Language*. https://github.com/input-output-hk/plutus-prototype/blob/master/docs/spec/Formal%20Specification%20of%20the%20Plutus%20Language%20-%20McAdams.pdf.

[8] Ralph C Merkle. "A certified digital signature". In: *Conference on the Theory and Application of Cryptology*. Springer. 1989, pp. 218–238.

[9] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008.

[10] Leonid Reyzin et al. "Improving Authenticated Dynamic Dictionaries, with Applications to Cryptocurrencies." In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 994.

[11] *Sony Global Education*. https://www.sonyged.com/.

[12] Melanie Swan. *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc., 2015.

[13] Gavin Wood. "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum Project Yellow Paper* 151 (2014).

A Appendix

A.1 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.2 Partially centralized educators

In the Section 3.4 we have concluded that the cost of the private block proof publication should depend on the size of the corresponding Merkle tree. This is done in order to scale spendings of different educators with amount of data they produce and store in their private blockchains.

But this solution has a disadvantage: the Witnesses are more incentivized to include proofs from large educators in public blocks rather than from small educators, as proofs from large educators contain more fees. If block size is limited, it may lead to delays of inclusion of small educators' proofs in the public blockchain.

In order to resolve this problem, small educators can use trusted third-party services (e. g. `teachmeplease.com`) for interacting with Disciplina platform instead of running Educator nodes by themselves. But this means that third-party service has access to all the educator's data, including marks and assignments of her students, and also receives all the revenue for trading this data. Some small educators might find this option unacceptable.

Therefore, we propose a mechanism of *educator pools*, which allow small educators to delegate block proof publishing to a third party in a partially trustless way.

The idea is the following:

- Every small Educator still maintains her own small private chain
- When a small Educator forms a block in her private chain, she sends the block header to a third party called *pool manager* instead of publishing it directly to Witnesses. Another difference is that Educator should also send a separate signature for her *ATG delta* (if it's not empty).
- A *pool manager* collects block headers from Educators until total number of transactions in all Educator's blocks is more than some threshold K_{min}.
- Then pool manager builds a sized Merkle tree over the list of received Educator's block headers, forming a *second-level block* (Fig. 8). The header of second-level block gets published on the public blockchain. Instead of containing a single *ATG delta*, the header of this second-level block contains a list of separate signed *ATG deltas* of small educators. We assume that this approach would not create a problem of oversized block headers because Educators don't typically create and close courses very often, and an average number of ATG deltas in every single block header will stay small.

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

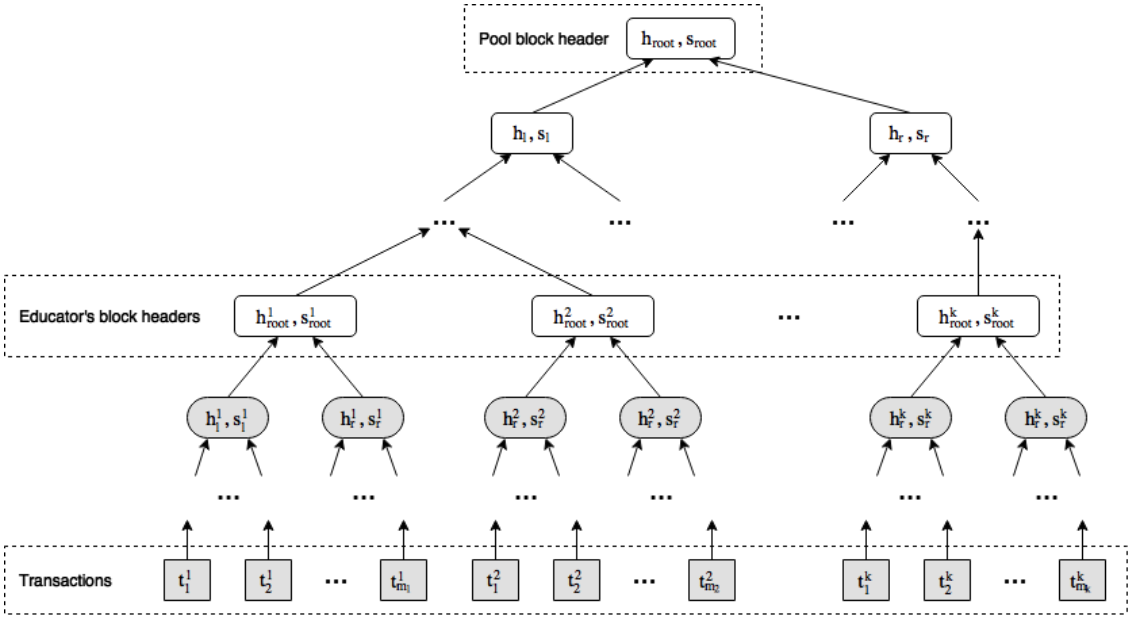

Figure 8: Two-level hierarchical sized Merkle tree for a block published by a pool manager

- After constructing a second-level block, pool manager sends each of the small Educators a path to their block headers in a second-level sized Merkle tree.
- Having this path, each Educator can construct a publicly verifiable proof for any transaction in her private block by simply concatenating this path with a path to transaction in a first-level Merkle tree.

For every processed block header small educator pays pool manager a fee calculated by this formula:

$$C_{pool}(B) = \alpha_{pool} + \beta_{pub} \cdot N_{tr}(B) \tag{12}$$

where β_{pub} is a network price coefficient from 6, and α_{pool} is a constant fee set by the pool manager.

If a pool manager sets such α_{pool} that $\alpha_{pool} < \alpha_{pub}$, but $\alpha_{pool} N > \alpha_{pub}$, then for every published second-level block a pool manager gains $\alpha_{pool} N - \alpha_{pub}$ coins, while every Educator in pool pays less for the block header publishing then if published directly to Witnesses. Therefore, every participant has an incentive to remain in the pool.

Note that a small Educator participating in the pool should slightly change the structure of his own chain. Every block header should contain a hash of previous one (as described in Section 3.4), but blocks of an Educator participating in the pool are published only as part of second-level pooled block. So the educator's block header should contain a header hash of a second-level block containing previous first-level block in educator's chain instead of header hash of previous first-level block.

We don't yet provide a fully trustless solution for pooling. Small Educators should trust a pool manager to provide correct second-level proofs of their blocks, and in theory an adversarial pool manager may cheat on Educators by not including their block headers in a published second-level block after she received the money. However, we assume that deceiving the small educators and losing them as a long-term clients is far less profitable than being honest. Nevertheless, we are currently working on a fully trustless pooling protocol based on a smart contract which ensures that a pool manager cannot deceive any Educator.

A.3 Smart-contract implementation

A.3.1 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.

A.3.2 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".

Each atomic step of execution has an invocation cost.

A.3.3 NIFs

NIF is an 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.

A.3.4 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(Sender-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 the 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 a transaction, the code invocation is assumed successful.

The `Gas` fee for transaction approval will be calculated as sum of costs for atomic actions performed.

A.3.5 Implementation of the data disclosure protocol using Robot account

We assume that we have 2 sides:

- Buyer
- Seller.

"Gas" below is the estimation of the operation cost. The name and the idea are taken from Ethereum. "Fee" is a forfeit to either side trying to deceive the opponent. "Sum" is the price of the data to be sold.

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

- `start-trade();`
- `accept-encrypted(encrypted-mroot, size);`
- `send-encrypted-secret(encrypted-secret);`
- `handshake();`
- `reject(sk, block, proof-enc);`
- `cancel(),`

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

This robot is initialized with the following data:

- Cost in tokens;
- Size of the encrypted data;
- Merkle tree root of encrypted data;
- Timeout, after which the account is destroyed;
- Sink, a person whom the remaining money will be send on timeout;
- Destructor, a procedure to invoke on timeout.

Here is the state machine of that Robot-account:

```
(0) Account has just been created.
  "start-trade" from Seller
    WHEN transferred amount = Fee
      Notify Buyer
      AND GOTO 1

(1) The trade is started.
  "accept-encrypted(encrypted-mroot, size)" from Buyer
    IF encrypted Merkle root and size are the same as initially set in Robot state,
      GOTO 2
    ELSE
      GOTO 8

  Timeout!
    GOTO 8

(2) Data was transferred. Seller encrypts a session key with the Buyer's public key.
  "send-encrypted-secret(encrypted-secret)" from Seller
  Notify Buyer

(3) Off the band, the symmetric key is sent.
  "reject(sk, block, encrypted-proof)" from Buyer
```

```
    GOTO 4

  "handshake" from Buyer
    GOTO 5

  Timeout!
    GOTO 6

(4) Arbitration.
  The robot performs check.
  If it finds that Buyer is right (block invalidates proofs OR the predicate fails),
    GOTO 7
  else if Seller is right
    GOTO 6

(5) Cleanup I.
  (Sum + Fee) is sent to Seller.
  GOTO 7

(6) Cleanup II.
  All remaining money is sent to Seller.
  The account is closed.

(7) Cleanup III.
  All remaining money is sent to Buyer.
  The account is closed.

(8) Cleanup VI.
  Fee is sent to Seller.
  GOTO 7
```

A.3.6 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.
- Robot, which is a smart account created by the Buyer.

We will consider three cases:

- Seller sends nothing at all;
- Seller tried to send Buyer encrypted garbage instead of data;
- Buyer tried to blame Seller in giving her invalid data with data being completely valid (in terms of proofs and predicate).

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

- The Buyer formulates a predicate to check that data corresponds its description OR uses universal truth as predicate.
- The Buyer requests the data from the Seller.
- The Seller notifies the Buyer of the data price (500 tokens).
- The Buyer creates a Robot using the scheme above with 500 tokens and the predicate.

- The Seller accepts the trade and sends `start-trade()` to the Robot along with 20 tokens Fee.

    1. Seller sends badly sized data or the Merkle root of the encrypted data does not match:
        - The size or the Merkle root of the transferred data is invalid.
        - The trade is stalled until time is out.
        - On timeout the trade is reverted and money is returned to Buyer.
    2. Seller tries to send garbage or one of blocks decrypts to garbage:
        - Buyer finds that at least one block is invalid in either form.
        - She invokes `reject(sk, block, encrypted-proof)` to start arbitration.
        - Robot checks that block falsifies proofs and that Buyer was right.
        - Robot returns all remaining funds to Buyer.
    3. 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.
        - Robot performs `check-block-and-proof(block, proof)` and finds that Buyer was not right.
        - Then it sends all 520 tokens to Seller.
    4. Buyer receives the data and the key, but remains silent:
        - If the timeout has expired, 520 tokens are sent to Seller.