Kahramanmaras Sutcu Imam University
Journal of Engineering Sciences

Geliş Tarihi : 10.05.2024
Kabul Tarihi : 31.07.2024

Received Date : 10.05.2024
Accepted Date : 31.07.2024

A SECURE ONLINE EXAMINATION SYSTEM USING SMART CONTRACTS

AKILLI SÖZLEŞMELER KULLANIMIYLA GÜVENLİ BİR ÇEVRİMİÇİ SINAV SİSTEMİ

Ozgur OKSUZ[1] (ORCID: 0000-0001-5568-6116)

[1] Konya Technical University, Department of Software Engineering, Konya, Türkiye

*Sorumlu Yazar / Corresponding Author: Ozgur OKSUZ, ooksuz@ktun.edu.tr

ABSTRACT

Covid19 pandemic has affected many sectors including education. All types of schools (public and private) have started to provide online education systems to their students to prevent spreading the disease. The online education system has brought many advantages besides stopping the spread of the disease. The students have more time since the lectures are not in class, it reduces the cost for the students since they physically do not go to school, and it provides flexibility in the lectures that students decide their schedule to attend the classes. However, there are some challenges in the online education system. The system needs to be available for the students when they take exams, view their exam results, and upload their exams for assessment. Since the lectures and exams are taken online, the data is very crucial. The data contains sensitive information such as exams, answers, scores, name, date of birth, address, phone number, government identification number, etc. A traditional online education system has a centralized infrastructure governed and managed by a single entity. This results in the system having single-point-of-failure attacks. This paper proposes an online examination system based on a smart contract and a blockchain. The blockchain eliminates single-point-of-failure attacks. The teachers write questions and store them in the blockchain. Only authorized students retrieve the exams from the blockchain by using smart contracts. The students submit their completed exams at predefined times. Then, the teachers evaluate the students' exams and put the results into the blockchain. Their exam scores are protected from any unauthorized entities by encryption. Students can freely see and view their scores at any time. Then, the students can show their results to any other third parties (once they apply for an internship or job) that they have completed the courses successfully. The system also uses a decentralized storage (off-chain) system to eliminate scalability problems. Off-chain storage (InterPlanetary File System) stores students' exams, answers, and exam results while the corresponding content identifiers of the files are stored in the blockchain. The proposed system is resilient to malicious teachers who can manipulate the exam results. In addition, the proposed system also provides a method for dishonest students who can complain about their exam results. In other words, the proposed system solves any conflicts between entities.

Keywords: Blockchain, encryption, online-test, privacy, smart-contracts

ÖZET

Kovid19 salgını eğitim başta olmak üzere birçok sektörü etkiledi. Hastalığın yayılmasını önlemek amacıyla tüm okul türleri (kamu ve özel), öğrencilerine çevrimiçi eğitim sistemleri sunmaya başladı. Online eğitim sistemi hastalığın yayılmasını durdurmanın yanı sıra birçok avantajı da beraberinde getirdi. Derslerin sınıfta olmaması nedeniyle öğrencilere daha fazla zaman kalıyor, fiziki olarak okula gitmedikleri için öğrenciler için maliyet düşüyor ve öğrencilerin derse katılım programlarını kendilerinin belirlemesi derslerde esneklik sağlıyor. Ancak çevrimiçi eğitim sisteminde bazı zorluklar var. Öğrencilerin sınava girmeleri, sınav sonuçlarını görüntülemeleri ve sınavlarını değerlendirmek üzere yüklemeleri sırasında sistemin kullanılabilir olması gerekmektedir. Dersler ve sınavlar online olarak yapıldığı için veriler çok önemlidir. Veriler sınavlar, cevaplar, puanlar, isim, doğum tarihi, adres, telefon numarası, resmi kimlik numarası vb. gibi hassas bilgileri içerir. Geleneksel bir çevrimiçi eğitim sistemi, tek bir kuruluş tarafından yönetilen ve idare edilen merkezi bir altyapıya sahiptir. Bu, sistemin tek hata noktası saldırılarına sahip olmasına neden olur. Bu makale akıllı sözleşmeye ve blokzincirine dayalı bir çevrimiçi sınav sistemi önermektedir. Blokzincir, tek nokta başarısızlık saldırılarını ortadan kaldırır. Öğretmenler sorular yazar ve bunları blokzincir'de saklar. Akıllı sözleşmeleri kullanarak sınavları yalnızca yetkili öğrenciler blokzincir'den alır. Öğrenciler tamamladıkları sınavları önceden belirlenen zamanlarda teslim ederler. Daha sonra öğretmenler öğrencilerin sınavlarını değerlendirir ve sonuçları blokzincir'e koyar. Sınav puanları her türlü yetkisiz kişiden şifreleme yoluyla korunur. Öğrenciler puanlarını istedikleri zaman özgürce görebilir ve görüntüleyebilirler. Daha sonra öğrenciler, dersleri başarıyla tamamladıklarını (staj veya işe başvurduklarında) sonuçlarını diğer üçüncü taraflara gösterebilirler. Sistem ayrıca ölçeklenebilirlik sorunlarını ortadan kaldırmak için merkezi olmayan bir depolama (zincir dışı) sistemi kullanıyor. Zincir dışı depolama (InterPlanetary Dosya Sistemi), öğrencilerin sınavlarını, cevaplarını ve sınav sonuçlarını saklarken, dosyaların karşılık gelen içerik tanımlayıcıları da blokzincirinde depolanır. Önerilen sistem, sınav sonuçlarını manipüle edebilecek kötü niyetli öğretmenlere karşı dayanıklıdır. Ayrıca önerilen sistem, sınav sonuçları hakkında şikâyette bulunabilecek dürüst olmayan öğrencilere de bir yöntem sunmaktadır. Başka bir deyişle, önerilen sistem varlıklar arasındaki her türlü çatışmayı çözmektedir.

Anahtar Kelimeler: Blokzincir, şifreleme, çevrimiçi test, gizlilik, akıllı sözleşmeler

INTRODUCTION

At the end of 2019, people faced a global pandemic (COVID-19) that has been transmitted between people by breathing contaminated air. To prevent the spread of the disease, people keep their distance from others and avoid crowds. Many sectors like healthcare and finance have been affected. Education is one of these sectors. Schools (universities, middle schools, high schools) should have redesigned their education systems since COVID-19. The schools have started using the online education system. The students have had their education at home to prevent the spread of the disease. They have taken courses and had their exams online.

However, online education brings some disadvantages. The main disadvantages are protecting the privacy of the students' data, securing the system, making the online examination system available, and providing non-repudiation of the exam results. The schools need a computerized infrastructure for issuing exams and storing the exam results. Moreover, the system should allow entities to view and verify their exam results anytime and anywhere. The students may show their results to third parties once they apply for a job.

Furthermore, the students' data should not be leaked to unauthorized entities since it contains sensitive information about the students (name, date of birth, phone number, address, social security number, government identification number, exam files, exam results, etc). In addition, the system should be available for the students to see and verify their test results whenever they want. The data on the test system needs to be available, reliable, immutable, and transparent. In the system, some dishonest parties (teachers or students) might tamper with the students' data and take advantage of it. Thus, the test results should be verifiable to eliminate any fraudulent activity. Classical examination systems are fully centralized and governed by a single entity. So, the system is available for single-point-of-failure attacks. Once a trusted entity is compromised, the students' sensitive data will be leaked to untrusted parties.

Since the exam questions (classic and multiple choice) can include pictures, tables, and figures and can have a large number of questions, the exam file should not be stored directly in the blockchain. This results in scalability problems in the network since the users (students and teachers) have limited storage and computation sources. It should be a mechanism to solve the scalability problem in the online examination system.

This paper addresses the problems above by proposing a smart contract and a blockchain-based online examination system using a decentralized file storage protocol (*IPFS*). The proposed system uses blockchain technology as a building block to store every exam and the student’s results in the blockchain. Moreover, the system uses smart contracts for the users to access their exams and exam results (authorized access). The use of the blockchain allows the data not to be changed by anyone and is transparent. Moreover, the system will be resilient to single-point-of-failure attacks. Furthermore, the data will be available anytime and anywhere. The Hyperledger blockchain (Androulaki et al., 2018) can be used to deploy smart contracts and to allow users to access the exams and results by connecting the network using their credentials (usernames and passwords). The proposed system encrypts the sensitive information to preserve data privacy. The system provides a control mechanism to track the transactions and eliminates conflicts between the entities.

In addition, the proposed system uses the InterPlanetary File System (*IPFS*) to store the original exam files (large files). The reason why the original files are stored in the *IPFS*, not on the blockchain, is to eliminate the scalability problem. Since the original data is stored outside of the blockchain, the *IPFS* is known as an off-chain storage. The hashes of the original files (they are called content identifiers of these files) are stored on the blockchain (on-chain). It is called on-chain since the data is stored directly on the blockchain. The transactions (data) stored in both databases (on-chain and off-chain) are verifiable. The contribution of the proposed system has the following properties:

- The system uses blockchain technology as a building block to eliminate single-point-of-failure attacks that the classic online education system has. The data on the blockchain is immutable, transparent, and available anytime and anywhere.
- The smart contracts allow the students to access the exams and the results. Moreover, they are stored in the blockchain.
- The students' sensitive data is protected by encryption so that any unauthorized entities cannot see the results.
- Using the blockchain provides non-repudiation for the entities. Students cannot deny or complain about their exam scores. The system solves the conflict between entities (teachers and students) about the students' exam results when they are dishonest. All data (test and test results) on the blockchain is verifiable.
- The proposed system uses the *IPFS* protocol to provide scalability.

The outline of the paper is as follows: Section 2 provides the recent studies about online examination systems. Section 3 gives some definitions that the proposed system uses. Section 4 presents the architecture and the workflow of the system. In section 5, the proposed system is explained in detail. In section 6, the complexity analysis of the system is given. Section 7 gives the security analysis of the proposed system. Section 8 discusses some limitations of the proposed system. The conclusion and future work are given in section 9.

RELATED WORK

Several studies proposed an online examination system. The study (Pee et al., 2019) proposed a smart contract-based online test system. They used private blockchain and attribute-based encryption to provide privacy for the students. However, it does not explain how the entities interact with the smart contract for a reliable, secure, and transparent system. Moreover, it does not examine the conflict between entities. Another work (Tentea & Ionescu, 2019) proposed an online testing system that provides a solution for non-repudiation, storing test results, and result tempering. However, the authors do not focus on the privacy of the students. In study (Kim & Huh, 2020) proposed an online cheating test system to catch irregularities using artificial intelligence. Another work (Jain et al., 2021) proposed a smart contract-based examination system in which students first pay their examination fee to authenticate themselves. The work compares cloud-based solutions to blockchain-based solutions for online examination systems. However, the work does not give details of the smart contract and interactions between entities and the smart contracts. The study (Jain et al., 2021) proposed an online examination system using a private blockchain and IPFS. However, (Kapse et al., 2022) does not have a proper adversarial model to prevent any fraudulent behaviour of the entities.

In addition, the study does not give details of the smart contract. A recent work (Sattar, 2023) proposed a blockchain-based secure online examination system. The authors (Sattar, 2023) focused on secure login to the examination system. The system uses IP address and face detection and voice recognition-based login. The authors also focused on removing any potential cheating using artificial intelligence. However, their work does not have a proper adversarial model and is not based on smart contracts. Study (N et al., 2022) focused on preventing cheating in an online exam. They used a face detection mechanism to proctor the exam.

The work (Abdelsalam et al., 2024) proposed a model for improving online exam results based on blockchain. The authors focused on developing a blockchain-based system to provide a secure evaluation of the exams with integrity of the results. To allow the students to answer the questions directly on the blockchain, the authors developed a module that integrated with the Moodle learning management system. However, storing the questions and answers directly on the blockchain results in a scalability problem since the questions take up large space since they can have pictures, files, and tables. However, in their paper, there is no threat/trust model to capture all attacks.

(Li et al., 2019) proposed a blockchain-based e-learning system that uses two different blockchains (public and private). The public blockchain was used for storing e-learning data and managing credits. The private blockchain was used for managing grades. However, their system did not have any information about the threat or trust model.

Another work (Zhu & Cao, 2021) focused on a blockchain-based online examination system that uses a biometric authentication method for users. In this work, the data privacy of the entities was preserved using an attribute-based encryption scheme, and the examination data was stored in the distributed storage system. Since the examination of the data was encrypted in the distributed storage system, the entities accessed the data only if the entity had corresponding attributes. However, in their trust model, teachers were trusted. The authors did not focus on when the teachers were malicious and gave lower scores to the students. Moreover, attribute-based encryption uses bilinear paring that results in intensive computation.

(Tsai et al., 2022) presented a blockchain-based secure scoring mechanism for online learning that uses Ethereum (public) blockchain (Buterin, 2014). The system used a trusted administrator to address the conflict between students and teachers. Since the system uses a public blockchain, the administrator communicates with the teachers to give their student lists. Therefore, it eliminates non-registered students from taking the exams. To protect the privacy of the students, the system used *RSA* (Rivest et al., 1978) encryption. The students encrypted their answers before putting them into the blockchain. The system did not use a distributed storage system. This results in a scalability problem in the network since the questions can contain pictures, tables, and figures that take up large space. In addition, the system does not examine cases in which the teacher gives lower scores to the students than they are supposed to have.

Another study (Manawar, 2023) proposed a blockchain-based online examination system focused on some flaws in the exams such as cheating, fake identities, common questions, secure publication, and multiple devices. To address these problems the study proposed an AI and blockchain systems. However, the study does not focus on malicious teachers to tamper with the students' exam results. The paper does not have a formal trust model.

(HAM et al., 2021) proposed a blockchain and smart contract based examination, transcript, and certificate system. The system is provided with a light blockchain system composed of several modules. However, the study does not have a proper threat model. The study does not focus on malicious teachers to tamper with the examination results of the students. Another disadvantage is that the examination system only supports multiple-choice questions.

The proposed system in this paper differs from the above studies. The proposed system is based on providing data privacy for students so that any unauthorized entity cannot get students' scores and answer files. Moreover, the proposed protocol solves the conflicts between the students and the teachers once entities are dishonest. Furthermore, the proposed study uses a distributed, peer-to-peer storage service to store students' large files. The large files are not stored in the blockchain to provide scalability. The proposed system does not focus on preventing any cheating in online tests.

DEFINITIONS

Blockchain

The blockchain is a peer-to-peer and decentralized network that consists of blocks. Each block contains a bunch of transactions that are infeasible to tamper. In other words, the data in any block is infeasible to change. Once a transaction is issued, it is sent to other nodes for verification. After verification, the transactions are formed into a block, and the block is added to the blockchain. This process is called a consensus. The blocks are attached by cryptographic algorithms (hash and signature). If any transaction is changed in a block, the honest nodes will catch and discard it. There are three types of blockchain: Public, Permissioned, and Private. In a public blockchain, anyone can join the network without any restrictions. In addition, the decentralization level is higher than private and permissioned blockchains. Examples of public blockchains are *Bitcoin* (Nakamoto, 2008) and *Ethereum* (Buterin, 2014). Private blockchain is less decentralized (Viriyasitavat & Hoonsopon, 2019) and is used in business, industrial, and government use cases (Swan et al., 2018). Permissioned blockchains are between public and private blockchains (Viriyasitavat & Hoonsopon, 2019). In a permissioned blockchain, only permitted nodes can join the network, validate transactions, and create blocks. Fig. 1 demonstrates a blockchain and its structure. The block consists of two parts: Header and Body. The header consists of a block number, a previous block hash, a Merkle tree root, and a timestamp. The body consists of transactions, a current block hash, and a signature of the current block hash. The transactions are formed as the leaf nodes on the tree. A *Merkle* root is the hash of the hashes of all transactions in the block.

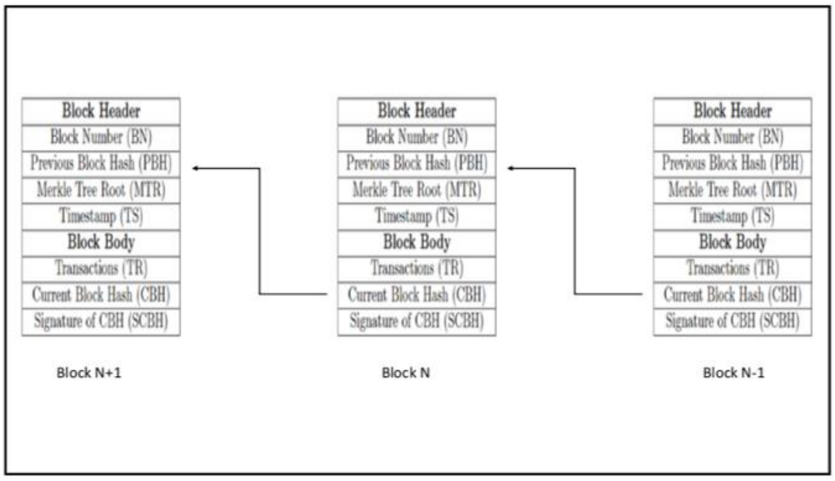

Figure 1. Blockchain Structure

Smart Contracts

Smart contracts are programs that consist of codes. These programs are automatically executed if pre-conditions are met. Once smart contracts are created, they are deployed. The codes are deterministic and consist of functions and attributes. Users interact with smart contracts using functions. To use the smart contract's function, the users pass some parameters to it and get the output of it. Each function has some requirements to be executed. This is called the predetermined conditions of the smart contracts. Once a user satisfies the preconditions, the function gives the output. The smart contracts are stored on the blockchain and nobody controls them. Each smart contract provides the following information: how users can interact with it, at what times, and what inputs result in what outputs. There is a cost for executing a smart contract and it is measured by gas. The cost of the execution gets higher once the smart contract gets complicated. In other words, the cost of the execution depends on the complexity of the smart contract's functions and its inputs. There are several platforms where smart contracts are deployed. The most used ones are Ethereum (Buterin, 2014) and Hyperledger (Androulaki et al., 2018).

Encryption

Encryption is simply an algorithm. An encryption algorithm has three phases: *KeyGeneration*, *Encryption*, and *Decryption*. In the key generation phase, a trusted party generates the public parameters, and each user chooses their secret and public keys. In the encryption phase, the user encrypts a message with the recipient's public key and sends it to the recipient. The recipient then uses their secret key to decrypt the received ciphertext which happens in the last phase.

In this work, teachers and students use *ElGamal* encryption (ElGamal, 1985) scheme to protect their data privacy since it is a *CPA (chosen plaintext attack)-secure* encryption scheme.

ElGamal encryption algorithm is defined as follows:

Let G be an elliptic curve group of prime order q with a generator P. For secret key, each user chooses a random $k_t \in Z_q$ (k_s) and sets its secret key as $sk_t = k_t$ $(sk_s = k_s)$. The index represents the owner of the key which is teacher (student). Then, it sets their public key as $pk_t = k_tP \in G$ $(pk_s = k_sP \in G)$. In the encryption phase, to encrypt a message $m \in G$, the user chooses a random r_t and computes $(C_1, C_2) = (r_tP, m{+}r_tpk_s)$. The ciphertext consists of two elements in the group. In the decryption phase, to decrypt the message, the receiver computes $(C_2 - k_sC_1)$ which leads to message m.

The security of *ElGamal* (ElGamal, 1985) encryption is based on the hardness of *Decisional Diffie Hellman (DDH)* assumption.

Signatures

A signature algorithm consists of three phases: *KeyGeneration*, *Sign*, and *Verify*. In the key generation phase, a trusted party generates the system's public parameters. In addition, each user chooses their signature secret and public (verification) keys. The *Sign* algorithm takes a message, and the signer's secret key outputs a signature. The *Verify* algorithm takes the signer's public key, the message, and the signature. Then, it outputs 1 if the signature is formed by the signer's secret key and the message. Otherwise, it outputs 0.

In this work, the proposed system uses *Elliptic Curve Digital Signature Algorithm* (*ECDSA*) (Johnson, 2001). *ECDSA* provides the same level of security as other signature algorithms using shorter key lengths.

ECDSA algorithm is defined as follows:

Let G be an elliptic curve group of prime order q with a generator P (base point). Each user chooses a random $l_t \in Z_q$ (l_s) and sets its secret key as $ssk_t = l_t$ $(ssk_s = l_s)$. The index represents the owner of the key which is teacher (student). Then, it sets their public (verification) key as

$$spk_t = l_tP \in G\ (spk_s = l_sP). \tag{1}$$

In the $Sign$ phase, to sign a message m, a signer chooses a random r_t and computes

$$(x_R, y_R) = R = r_tP. \tag{2}$$

The signer computes

$$s = h(m) + l_tx_R. \tag{3}$$

The signature is $S = (x_R, s)$. In the verification step ($Verify$), the recipient computes

$$(x'_R, y'_R) = (h(m)s^{-1})P + x_Rs^{-1}spk_t \tag{4}$$

If $x_R = x'_R$, the signature is valid. Otherwise, the signature is not valid.

The security of $ECDSA$ is based the hardness of *Elliptic Curve Discrete Logarithm Problem (ECDLP).*

Consensus

A consensus mechanism needs to be provided in a blockchain system to form verified transactions in a block. Once a transaction is issued, this transaction is sent to the other entities for verification. Once the majority of entities check the validity of the transactions, then they are added to the blockchain as a block. Several studies proposed different consensus algorithms like *Proof of Work* (*PoW*), *Proof of Stake* (*PoS*), *Delegatable Proof of Stake* (*DPoS*), *Byzantine Fault Tolerance* (*BFT*), and *Practical Byzantine Fault Tolerance* (*PBFT*).

In this work, we use *PBFT* consensus protocol for the nodes to agree on the transactions. In *PBFT* consensus, the entities are a leader (master/primary), a client, and replica nodes. There are *3f+1* nodes and at most *f* of them are malicious (Byzantine). The protocol is initiated by a client by sending a request to the leader node. Then, the nodes continue with the phases of it until there is an agreement among them. Then the new record is added to the blockchain. The protocol consists of 3 phases: pre-prepare, prepare, and commit. In the pre-prepare phase, once the leader (master) node receives the request, it sends pre-prepared message to all other nodes (replicas). In the prepare phase, all other nodes check the messages and share their agreements or disagreements with each other and the leader. If a node gets at least *2f+1* prepare messages from others, then it verifies them and sends commit message to all other nodes. Then, if a node receives *2f+1* commit messages, it verifies the correctness of the messages. If so, the node sends a reply message to the client that it has *2f+1* verified committed messages. If the client receives *f+1* identical replay messages from others, it will know that the agreement has been made.

InterPlanetary File System (IPFS)

IPFS (Benet, 2014) is a distributed and (Peer-to-Peer) *P2P* system used to share and store large files. Once a user sends its file to the *IPFS* for storing, the system generates a unique hash for the content. Then, this content identifier is given to the user. The content identifier is for retrieving the original file from the *IPFS*. Moreover, it is also used for the integrity of the file in the *IPFS*. This addressing mechanism is known as content-based addressing.

THE ARCHITECTURE AND THE WORKFLOW OF THE PROPOSED SYSTEM

The architecture of the proposed system is illustrated in Fig. 2. In the figure, the students take exams, teachers prepare exams for students, and a certificate authority (*CA*) sets system public parameters for a hash function, encryption, signature algorithms, and assigns two secret/public key pairs (for encryption and signature algorithms) to the users. Moreover, it registers entities to the system. The registration can be done by giving the certificates to their public keys. Then, the certificate authority sends these public keys to everyone. The workflow of the system (Fig. 2) is as follows:

1. Students and teachers get secret/public keys from the certificate authority.
2. The teacher writes the exam questions and encrypts the exam with the student's public keys. Then, it sends the encrypted documents to the *IPFS* network. The network returns the hash of the file (content identifier) to the teacher.
3. The teacher issues a smart contract. It then sends it to the other peers for validation (consensus).
4. Once the transaction is valid, the teacher puts the smart contract into the blockchain.
5. The student sees the smart contract and retrieves the hash of the encrypted file by executing the function in the smart contract.
6. The student retrieves the encrypted exam from the *IPFS* by using the content identifier.
7. Once the student finishes the exam, it sends the encrypted answers to the *IPFS* and gets the hash of its encrypted answers (content identifier).
8. The student then issues a transaction by executing the function in the smart contract.
9. The teacher verifies the transaction and sends the transaction to its peers to form a block.
10. After consensus, the teacher puts the transaction into the blockchain.
11. Once the teacher gets the ciphertext of the student's exam from the *IPFS*, the teacher decrypts it and grades the exam.
12. The teacher puts the encrypted exam result into the *IPFS*. It then returns the hash of the encrypted exam result.
13. The teacher issues a transaction by executing a function in the smart contract. Then, it sends it to its peers.
14. After consensus, the teacher puts it into the blockchain.
15. The student retrieves the hash of the file (content identifier) from the blockchain.
16. The student retrieves the encrypted exam result from the database and decrypts it. In the end, the student learns its exam result.

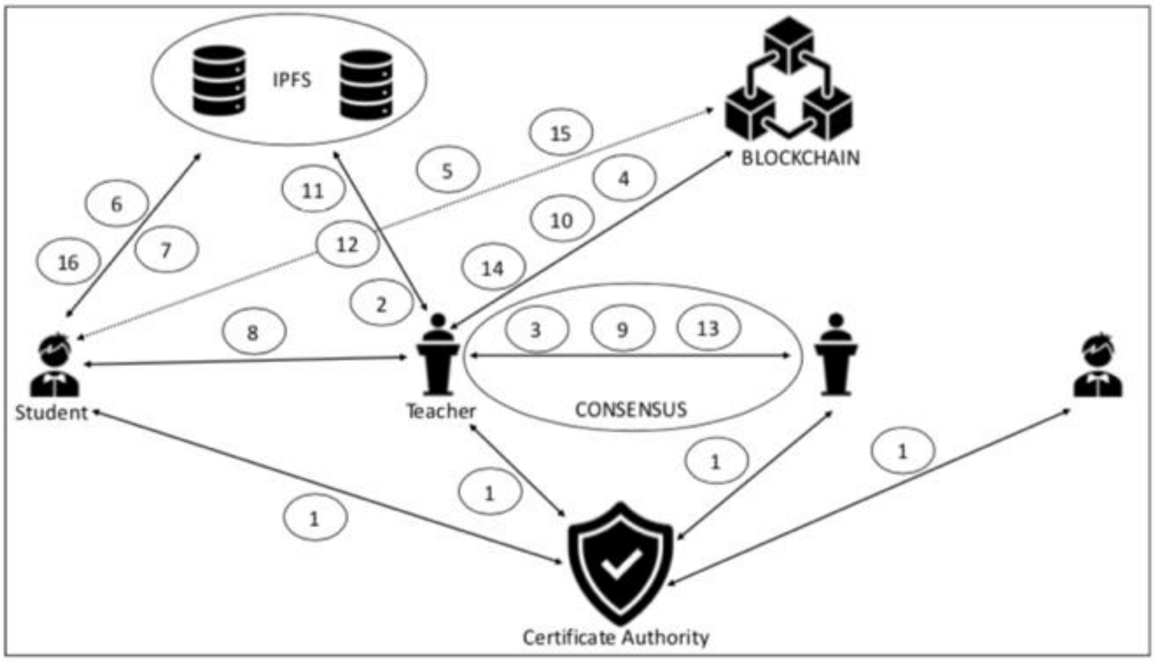

Figure 1. The Architecture and the Workflow of the Proposed System

In the proposed system, the consensus phase is as follows: each node is a teacher. Once the teacher creates its exam (the smart contract) for the students, it acts as the client. The teacher starts the consensus, issues transactions for its exam, and sends them to the leader (master/primary) node. The primary node checks if the transactions are valid, forms a block, and sends it to the other teachers for validation. After the *PBFT* consensus as in the *Consensus* section, the leader node adds the block to the blockchain.

Once the students issue transactions for submitting their answers as the transactions by calling corresponding functions, the corresponding teacher acts as a client. It collects and sends the transactions to the leader node to start the consensus. The leader node forms a block (after validation) with them and sends it to other nodes for validation. After the consensus, the block is added to the blockchain by the leader node (teacher).

Threat Model

In this paper, Certificate Authority (CA) is fully trusted and can be the president of the school/college/university that everyone relies on. Students can be dishonest and can argue with teachers for their grades. They can deny their answers to gain higher scores. Teachers can be malicious and can manipulate or change the exam results. An online examination system should satisfy the following properties:

1. Students' data privacy should be preserved. The system should protect students' data (name, date of birth, government identification number, address, exam results (grades), and answer sheets) from dishonest entities. This entity could be an insider (student, teacher) or any outsider of the system.
2. Students can not deny their exam results and answer sheets to get higher scores.
3. Teachers can be malicious and give students lower scores. The system should prevent dishonest teachers from giving lower scores to students.
4. Students should be able to verify their scores based on the answers.

THE PROPOSED SYSTEM

Since the entities should be known in advance, we use permissioned blockchain in the proposed system. We use Hyperledger fabric (Androulaki et al., 2018) platform to deploy smart contracts since it is permissioned blockchain. The certificate authority chooses system public parameters such as a hash function (SHA256),

$$h: \{0,1\}^* \rightarrow \{0,1\}^{\lambda} \quad (5)$$

(λ = 256 bits), a signature scheme *ECDSA* and an encryption scheme *ElGamal* for the system. For secure encryption, the key size of *ElGamal* should be 1024 bits (at least). Each user (teacher and student) gets a secret/public key pair for signature and a secret/public key pair for encryption from the *CA* via a secure channel. The use of the *CA* is to bind each user's public key to a user identity to authenticate its public key. The keys are chosen by the *CA* because the *CA* (president) needs to execute some functions on behalf of the teachers and the students to solve any conflicts between them.

Once a transaction happens, Teachers are the validating peers in the consensus. *PBFT* consensus is used to validate transactions. In this case, at least $\frac{2}{3}$ of the total teachers should be honest in the system.

The teacher writes test questions, encrypts them, and puts them into the *IPFS*. Then, it gets the hash of the encrypted exam. It starts writing smart contract's functions and sets its parameters. The smart contract's functions are *CreateContract, DeployTest, GetTest, SubmitAnswer, SendAnswerKey, EvaluateTest, SendScore,* and *RetrieveScore*.

CreateContract, DeployTest, SendAnswerKey, EvaluateTest, and *SendScore* functions are done by the teacher while *GetTest, SubmitAnswer,* and *RetrieveScore* functions are executed by the students.

CreateContract function is used to initialize the contract parameters by the teacher. It takes the identity of the exam, public key of a teacher, the student set that contains all students' addresses, the duration of the exam, a timestamp, and a signature of these values. The function outputs the address of the contract. The users use the address of the contract to interact with it. In this phase, the teacher also initializes some parameters. There are 4 lists used in the contract to keep track of the student's data. These lists are data structures stored in the blockchain (contract). Table 1 shows the *CreateContract* function.

Table 1. CreateContract Function

```
**Initialize**
$L1 = [], L2 = [][], \text{L3}=[][], L4 = [], TestDuration = 0, StSet = []$.
**CreateContract**$(Test_{id}, pk_{t_j}, StSet, TestDuration, ts_c, Sig_{l_{t_j}}(h(Test_{id}, StSet, TestDuration, ts_c)))$.
1. check if $add_{t_j} = HA(pk_{t_j}) \in TeSet$.
2. check signature if it is valid.
3. if all is valid returns address of the contract.
```

CreateContract function does some checks before outputting the contract address. First of all, it checks if the creator is authorized to create the smart contract by computing the address of the creator using the hash the public key. Then it verifies if it is in the teachers' set in step 1. It checks if the creator's signature is valid in step-2. Then, it returns the address of the contract in step-3.

DeployTest function takes the identity of the exam, public key of a teacher, address of a student, the hash of the exam that consists of questions, the content identifier of the exam, a time stamp, and a signature of these values. The identity of the exam consists of the course subject, number, year, and term of the test. For example, *MATH-101-2023-1* shows the subject, number, year, and term in order. The teacher executes this function multiple times since it encrypts the same exam with different public keys. The hash of the encrypted test (exam) is represented by the content identifier (*cid*). Moreover, the content identifiers and hash of the plain exams of the students are stored in a list.

Table 2. DeployTest Function

```
**DeployTest**$(Test_{id}, pk_{t_j}, add_{s_i}, h(test_{s_i}), cid_{s_i}, ts_d, Sig_{l_{t_j}}(h(Test_{id}, add_{s_i}, h(test_{s_i}), cid_{s_i}, ts_d)))$.
4. check if $add_{t_j} \in TeSet$.
5. check if $add_{s_i} \in StSet$.
6. check if the signature is valid.
7. if all checks are valid, it outputs $L1[add_{s_i}] = (cid_{s_i}, h(test_{s_i}))$
```

DeployTest function does some checks before giving any output. First, it checks if the computed address is in the teachers' set by the given public key as a parameter in step-4. Moreover, it checks if the address of the student is in the student's address set in step-5. Furthermore, it checks if the caller's signature is valid in step-6. As output, it maintains a list to store each student's data as a key-value pair in step-7. Tab. 2 shows the *DeployTest* function's properties.

GetTest function takes the identity of the exam, public key of a student, a timestamp, and a signature of these values, and outputs hash of the exam and the content identifier of the encrypted exam. Then, the student retrieves the encrypted exam from the *IPFS* using the hash of the encrypted exam. The student decrypts the ciphertext and receives the exam as the plaintext. Once the student finishes the test on time, it sends the encrypted answer to the *IPFS*. Then, it gets the content identifier of the encrypted file. The student also computes the hash of the completed exam (answers).

Table 3. GetTest Function

```
**GetTest**$(Test_{id}, pk_{s_i}, ts_g, Sig_{l_{s_i}}(h(Test_{id}, ts_g)))$.
8. check if $HA(pk_{s_i}) = add_{s_i} \in StSet$.
9. check if the signature is valid.
10. if all checks are valid, it returns $(cid_{s_i}, h(test_{s_i}))$.
```

GetTest function does some checks before giving any output. First, it checks if the computed address is in the students' set by the given public key as a parameter in step-8. Then, it checks if the caller's signature is valid in step-9. It outputs the content identifier of the encrypted exam and the hash of the plain exam in step-10. Tab. 3 shows the *GetTest* function's properties.

SubmitAnswer function takes the identity of the exam, public key of a student, the hash of the student answer file, the content identifier of the file, a timestamp, and a signature of these values. It then outputs a list that stores hash of student's answers for integrity and the content identifier to the file (for each student) in the *IPFS*. This function is called by multiple times (the number of students). Tab. 4 shows the *SubmitAnswer* function's properties.

Table 4. SubmitAnswer Function

```
**SubmitAnswer**$(Test_{id}, pk_{s_i}, h(answer_{s_i}), cid'_{s_i}, ts_a, Sig_{l_{s_i}}(h(Test_{id}, h(answer_{s_i}), cid_{s_i}, ts_a)))$.
11. check if $add_{s_i} \in StSet$.
12. check if $(ts_a \leq ts_g + ExamDuration)$.
13. check if the signature is valid.
14. if all checks are valid, it outputs $L2[add_{t_j}][i] = (cid'_{s_i}, h(answer_{s_i}))$.
```

SubmitAnswer function does some checks before giving any output. In step-11, it checks if the address of the student is in the student's set. In step-12, it checks if the transaction time is less than or equal to the exam duration plus the time when the student gets the exam (*GetTest*). In step-13, it checks if the student's signature is valid. Then, it maintains another list to store students' data as a key-value pair (step-14).

SendAnswerKey function takes the identity of the exam, public key of a teacher, address of a student, the hash of the answer key file, the hash of the encrypted answer key file *(cid)*, a timestamp, and a signature of these values. This function is executed for every student. Tab. 5 shows the *SendAnswerKey* function's properties.

Table 5. SendAnswerKey Function

```
**SendAnswerKey**$(Test_{id}, pk_{t_j}, add_{s_i}, h(AnswerKey), cid''_{s_i}, ts_{sak}, Sig_{l_{t_j}}(h(Test_{id}, add_{s_i}, h(AnswerKey), cid''_{s_i}, ts_{sak})))$.
15. check if $add_{s_i} \in StSet$.
16. check if $add_{t_j} \in TeSet$.
17. check if the signature is valid.
18. if all checks are valid, it outputs $L3[add_{s_i}][i] = (cid''_{s_i}, h(AnswerKey))$.
```

SendAnswerKey function checks if the addresses of the students and the teachers are in the corresponding sets (step-15 and step-16). In step-17, it checks the signature of the teacher if it is valid. Then, it maintains a list to store the content identifier of the encrypted answer key file and the hash of the answer key for each student as a key-value pair in step-18.

GetStudentResult function takes the identity of the exam, public key of a teacher, a timestamp, and a signature of these values and outputs the content identifier and the hash of the student answer. The teacher then retrieves the encrypted answers of the students from the *IPFS* using the content identifiers. The teacher grades the exams, encrypts the exam scores, and puts them into the *IPFS*. The *IPFS* returns the content identifiers for the encrypted exam scores of the students. Tab. 6 shows *GetStudentResult* function's properties.

Table 6. GetStudentResult Function

```
**GetStudentResult**$(Test_{id}, pk_{t_j}, ts_r, Sig_{l_{t_j}}(h(Test_{id}, ts_r)))$.
19. check if $add_{t_j} \in TeSet$.
20. check if the signature is valid.
21. if all checks are valid, it outputs $(cid'_{s_i}, h(answer_{s_i}))$.
```

GetStudentResult function checks if the address of the teacher is in the teachers' set in step-19. It checks if the signature is valid in step-20. It outputs the content identifier of the student's completed encrypted exam and the hash of the student's completed plain exam in step-21.

SendScore function takes the identity of the exam, public key of a teacher, a student address, the hash of the encrypted exam result of the student, the hash of the student's score, a timestamp, and a signature of these values. Tab. 7 shows the *SendScore* function's properties.

Table 7. SendScore Function

```
**SendScore**$(Test_{id}, pk_{t_j}, add_{s_i}, cid'''_{s_i}, h(score_{s_i}), ts_{ss}, Sig_{l_{t_j}}(h(Test_{id}, cid'''_{s_i}, add_{s_i}, h(score_{s_i}), ts_{ss})))$.
22. check if $add_{t_j} \in TeSet$.
23. check if the signature is valid.
24. if all checks are valid, it outputs $L4[add_{s_i}] = (cid'''_{s_i}, h(score_{s_i})$.
```

In Tab.7, *SendScore* function checks if the address of the teacher is in the teachers' set in step-22. In step-23, it checks if the teacher's signature is valid. Then, it maintains a list to store each student's plain exam score and the content identifier of the encrypted score as a key-value pair in step-24.

RetrieveScore takes the identity of the exam, public key of a student, a timestamp, and a signature of these values, and outputs a content identifier to retrieve the encrypted score and a content identifier to retrieve the encrypted answer key file from the *IPFS*. Moreover, it also outputs the hash of the answer key file and the hash of the score for integrity. Tab. 8 shows the *RetrieveScore* algorithm.

Table 8. RetrieveScore Function

```
**RetrieveScore**$(Test_{id}, pk_{s_i}, ts_{rs}, Sig_{l_{s_i}}(h(Test_{id}, ts_{rs})))$.
25. check if $add_{s_i} \in StSet$.
26. check if the signature is valid.
27. if all checks are valid, it outputs $(cid''_{s_i}, cid'''_{s_i}, h(score_{s_i}), h(AnswerKey)$.
```

RetrieveScore function checks if the student's address is in the student's address list in step-25. Then, it checks if the student's signature is valid. It outputs the content identifier of the encrypted score of the student and the content identifier of the encrypted answer key of the exam, the hash of the plain score of the student, and the hash of the plain answer key in step-27.

COMPLEXITY ANALYSIS

In this section, we analyze the proposed protocol's complexity. Our analysis is based on m number of exams (smart contracts) that a teacher creates and a student takes.

Each teacher executes each smart contract five times: *CreateContract*, *DeployTest*, *SendAnswerKey*, *GetStudentResult*, *SendScore*. Each student executes each smart contract three times: *GetTest*, *SubmitAnswer*, and *RetrieveScore*.

Tables 9, 10, and 11 show the required storage, communication, and computational complexities for each user. Each teacher and student passes some parameters based on the function. We assume the *ECDSA* secret key (*sk*) is 256 bits, the public key (*pk*) is 512 bits, and the signature is 512 bits. Moreover, the identity of a test ($Test_{id}$) is 128 bits, the timestamp (*ts*) is 128 bits, the address (*add*) is 160 bits which is the output of *HA* (hash function).

$$HA: G \rightarrow \{0,1\}^{160} \tag{6}$$

TestDuration is 128 bits, and each content identifiers (*cid*) is 128 bits.

In Tab. 9, *CC* stands for *CreateContract*, *DT* stands for *DeployTest*, *GT* stands for *GetTest*, *SA* stands for *SubmitAnswer*, *SAK* stands for *SendAnswerKey*, *GSR* stands for *GetStudentResult*, *SS* stands for *SendScore*, *RS* stands for *RetrieveScore* functions. Moreover, n is the number of students that they take each test, $x = 1696$, $y = 1280$, $z = 1536$ bits. In Tab. 10, X_Y represents function Y is used X times, where

$$h: \{0,1\}^* \rightarrow \{0,1\}^{256} \tag{7}$$

Sig is signature, Enc is encryption, Dec is decryption, and SMC is smart contract creation algorithms. In Tab. 11, $X \leftrightarrow Y$ stands for the communication complexity between entity X and Y, $y' = 1024$ bits, CA is the certificate authority, T is teacher, and S is student.

Table 9. Communication Complexity Between Users and Smart Contract

	CC	DT	GT	SA	SAK	GSR	SS	RS
Student	–	–	$yO(m)$	$zO(m)$	–	–	–	$yO(m)$
Teacher	$\|CC\|O(m)$	$xO(nm)$	-	-	$xO(nm)$	$yO(nm)$	$xO(nm)$	–

Table 10. Computational and Storage Complexity of the System

	Storage	Computation
Student	$O(1)$	$O(m)_{Enc} + O(m)_{Dec} + O(m)_h + O(m)_{Sig}$
Teacher	$O(n)$	$O(m)_{SMC} + O(mn)_{Enc} + O(mn)_{Dec} + O(mn)_h + O(mn)_{Sig}$

Table 11. Communication Complexity Between Entities

	CA ↔ S	CA ↔ T	S ↔ IPFS	T ↔ IPFS
Student	y'	–	$4y'm$	–
Teacher	–	$y'n$	–	$4y'mn$

SECURITY ANALYSIS

In this section, we analyze the security properties of the proposed system that satisfy all the requirements mentioned in the *Threat Model* subsection.

1. The students view the contract once it is deployed to the blockchain. For each student, the teacher encrypts the exam and the exam result. The original files (exam and exam results) are stored in the *IPFS*, while the content identifiers of them are stored in the blockchain. The student retrieves the content identifiers by executing *GetTest* and *RetrieveScore* functions. Then, it gets the encrypted exam and the result from the *IPFS*. The student decrypts them and gets the plaintexts. Since the data is encrypted using the student's public key, only the owner of the secret key decrypts the ciphertexts. Thus, any unauthorized entity cannot get the plaintexts of the results and the answer sheets. Thus, the data privacy of the students is preserved and this step is satisfied.
2. Each student submits their encrypted answers with the hash of their plain answers by executing *SubmitAnswer* function. The outputs of this function are a content identifier of the encrypted document and a hash of the student's unencrypted answer $(cid'_{s_i}, h(answer_{s_i}))$. The teacher decrypts the encrypted exam, gets the plaintext $answer'_{s_i}$ of the student, and computes the hash of it. Then, it checks if the result $h(answer'_{s_i})$ equals $h(answer_{s_i})$. If the equality does not hold, it can be
   A. The *IPFS* network changes the corresponding student's encrypted answer.

B. The student lies about his answers. The student uploads different files to the blockchain and the *IPFS* that do not match. The student uploads a different encrypted document to the *IPFS* and a different hash of its answer to the blockchain.
C. The teacher lies.

To decide which problem occurs above, the president executes *GetStudentResult* function on behalf of the teacher to retrieve $(cid'_{s_i}, h(answer_{s_i}))$. The president checks if cid'_{s_i} equals the hash of the encrypted answer stored in the *IPFS*. If not, the *IPFS* network changed the value. It applies to *A*. If these checks are valid, the president decrypts the encrypted student's answers using the teacher's decryption key and obtains $answer'_{s_i}$. Then, the president checks if $h(answer'_{s_i}) = h(answer_{s_i})$. If the equality does not hold, the student lies about his exam. It applies *B*. Otherwise, the teacher lies. It applies *C*.

3. Once student executes *RetrieveScore* function and retrieves $(cid''_{s_i},\ cid'''_{s_i},\ h(score_{s_i}), h(AnswerKey))$. Then, the student gets its score $score'_{s_i}$ using cid'''_{s_i}, and gets $AnswerKey'$ using cid''_{s_i}. Then, it checks if $h(AnswerKey) = h(AnswerKey')$ and $h(score_{s_i}) = h(score'_{s_i})$. If not the student can complain about its grade. This is due to
   A. The *IPFS* network can change the encryption of the designated teacher's encrypted answer key file and the student's encrypted score.
   B. The student can lie about his answers. The student uploads a different encrypted document to the *IPFS* and a hash of its answer to the blockchain.
   C. The teacher lies about the values. The untrusted teacher gives a lower score to the student.

To decide which problem occurs above, the president first retrieves all the values from the smart contract by calling corresponding functions in behalf of the student and the teacher. It uses *GetStudentResult* and *RetrieveScore* functions. From *GetStudentResult* function, it retrieves $(cid'_{s_i}, h(answer_{s_i}))$ and from *RetrieveScore* function, it gets $(cid''_{s_i},\ cid'''_{s_i},\ h(score_{s_i}), h(AnswerKey))$.

The president checks if cid'_{s_i} equals the hash of the student's encrypted answer like in step-2. If not, the *IPFS* network changed the value. It applies to *A*. Otherwise, the president checks if cid''_{s_i} equals the hash of the encrypted answer key. If not, the *IPFS* network changed the value. It applies *A*. Otherwise, the president checks if cid'''_{s_i} equals the hash of the encrypted score. If not, the *IPFS* network changed the value. It applies *A*. If these checks are valid which means that the *IPFS* did not tamper any data, the president decrypts the encrypted student's answers using the teacher's decryption key and obtains $answer'_{s_i}$. The president checks if $h(answer'_{s_i}) = h(answer_{s_i})$. If the equality does not hold, the student lies about his exam. It applies *B*. If the equality holds, the president decrypts the student's encrypted score and the teacher's encrypted answer key using their decryption keys. Then, it obtains $AnswerKey'$ and $score'_{s_i}$. It then checks if $h(AnswerKey) = h(AnswerKey')$. If not, the teacher lies (C). Otherwise, it grades the student's exam using $answer'_{s_i}$ and the answer key $AnswerKey'$. It gets the result of the student's exam $score''_{s_i}$. It checks if $h(score''_{s_i}) = h(score'_{s_i})$=$h(score_{s_i})$. If not, the teacher is dishonest. It applies to *C*. Otherwise, the student lies (*B*). Thus, the step is satisfied.

4. The students can check their answers and the answer key prepared by the teacher to verify their scores. The students execute *RetrieveScore* function to do this. This step is also satisfied.

A note that if a server (node) in the *IPFS* does temper any data (encrypted exam of the student, encrypted score of the student, encrypted answer of the student, encrypted answer key), then the corrupted *IPFS* node will not be used for the future queries by the teacher. In other words, the corrupted node will be blacklisted. The teacher should ask any other nodes in the *IPFS* to have the original file. To eliminate having corrupted files, each entity can run its own node in the *IPFS*. Another note is that we assume that the students and teachers do not change the *cid* values received from the *IPFS* once they issue transactions with them.

DISCUSSIONS

In this section, we give some limitations of the proposed scheme. In the proposed system, there is a method for checking the integrity of students' scores when a malicious teacher gives lower scores than the students should get. However, there is no verification method when a teacher gives a higher score than a student should have. If the student does not complain about it, this problem will not be detected. Usually, students complain about their grades when they get lower scores. One solution to prevent teachers from giving higher scores to the students, the scores, the answer keys, and the students' answers should be stored as plaintexts in the blockchain. Thus, anyone can check other students' scores based on their answers. In this case, the student's data privacy will not be protected.

Another limitation occurs when the teacher sends an encrypted exam to the *IPFS* network. Once the hash of the encrypted exam *cid* is received, the teacher puts it into the blockchain using *DeployTest* function. When students execute the *GetTest* function, retrieve *cid*s, and then get the encrypted exams from the *IPFS*. If the data in the *IPFS* is changed, this can be noticed by the students and the teacher. However, they cannot have the exam. Then, this can be fixed immediately by the teacher. The teacher should prepare the same or a different exam.

Another limitation is if there is a conflict between a student and a teacher regarding the student's grade. The president should be online to solve the situation. Once a student complains about his grade, the president comes into play to provide the correctness. Thus, the president should work on this to solve the conflict among them (the teacher, the student, and the *IPFS).*

A note that integrating blockchain into the education system provides very useful properties such as the transparency of the grades, availability of the system, immutability of the records, and decentralized system. However, there are some obstacles to this integration. The recent studies (Mohammad & Vargas, 2022), (Koshiry et al., 2023), and (Dwivedi & Vig, 2023) showed that there are some challenges to the integration. One of the main challenges is that blockchain technology is a new tool and the students and the teachers are not aware/familiar with it. They may lack knowledge about the blockchain. Thus, it will take some time to learn the blockchain technology for them before they use it. They need to get comprehensive training. This results in the integration brings with it some costs. Moreover, there should be enough equipment (software and hardware) and expertise (for secure deployment) to adopt this technology.

CONCLUSION AND FUTURE WORK

In this paper, we proposed an online examination (test) system based on smart contracts and a blockchain. In the system, users (students) get their exams in the blockchain by logging into the system using their credentials. Then they read the smart contract and provide information to take the exams. To tackle the scalability problem, the files (student exams) are not stored in the blockchain directly. The proposed system uses decentralized storage protocol (*IPFS*). The students get their tests and freely view their scores anytime and anywhere. In our system, the students' data (exam) privacy is preserved by encryption. Moreover, the system provides a method to disallow dishonest students to complain about their exam results. Furthermore, the teachers cannot manipulate students' grades to give lower scores.

As a future work, we implement our system to see its performance. Moreover, we would like to provide a privacy-preserving method when malicious teachers give students higher scores than they are supposed to have.

REFERENCES

Abdelsalam, M., Shokry, M., & Idrees, A. M. (2024). A Proposed Model for Improving the Reliability of Online Exam Results Using Blockchain. *IEEE Access, 12,* 7719-7733. DOI: 10.1109/ACCESS.2023.330499

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., et al. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. *Proceedings of the Thirteenth EuroSys Conference on - EuroSys '18* (pp. 1-15). New York, New York, USA: ACM Press. DOI: 10.1145/3190508.3190538

Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System. CoRR, vol. abs/1407.3561. Retrieved from http://arxiv.org/abs/1407.3561.

Buterin, V. (2014). Ethereum. Retrieved 8 2023, from http://bitcoin.org/bitcoin.pdf

Dwivedi, S., & Vig, S. (2023). Blockchain adoption in higher-education institutions in India: Identifying the main challenges. Cogent Education, 11(1). DOI: https://doi.org/10.1080/2331186X.2023.2292887

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory, 31(4), 469-472. DOI: 10.1109/TIT.1985.1057074.

HAM, D., MM, D., & RK, S. (2021). The future of university education: Examination, transcript, and certificate system using blockchain. Computer Applications in Engineering Education, 29, 1234-1256. DOI: https://doi.org/10.1002/cae.22381

Jain, A., Tripathi, A. K., Chandra, N., & and Chinnasamy, P. (2021). Smart contract enabled online examination system based in blockchain network. 2021 International Conference on Computer Communication and Informatics (ICCCI). IEEE. DOI: 10.1109/ICCCI50826.2021.9402420

Johnson, D. M. (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA). International Journal of Information Security, 36(63). DOI: https://doi.org/10.1007/s102070100002

Kapse, S., Umalkar, M., Gajbe, A., Vrudhula, K., & Gour, R. a. (2022). Blockchain Based Solution for Secured Transmission of Examination Paper. 2022 IEEE 2nd International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC). IEEE. DOI: 10.1109/iSSSC56467.2022.10051340

Kim, S. -K., & Huh, J. -H. (2020). Blockchain Agreement for Self-identification of Online Test Cheating: Improvement of Algorithm Performance. 2020 20th International Conference on Control, Automation and Systems (ICCAS), (pp. 1124-1133.). Busan, Korea (South). DOI: 10.23919/ICCAS50221.2020.9268400.

Koshiry, A. E., Eliwa, E., El-Hafeez, T. A., & Shams, M. Y. (2023). Unlocking the power of blockchain in education: An overview of innovations and outcomes. Blockchain: Research and Applications, 4(4), 100165. DOI: https://doi.org/10.1016/j.bcra.2023.100165

Li, C., Guo, J., Zhang, G., Wang, Y., Sun, Y., & R. Bie. (2019). A Blockchain System for E-Learning Assessment and Certification. International Conference on Smart Internet of Things (SmartIoT) (pp. 212-219). Tianjin, China: IEEE. DOI: 10.1109/SmartIoT.2019.00040

Manawar, A. (2023). An Innovative and Secure Platform for Leveraging the Blockchain Approach for Online Exams. Aptisi Transactions on Technopreneurship (ATT), 5(1), 99-108. DOI: https://doi.org/10.34306/att.v5i1.314

Mohammad, A., & Vargas, S. (2022). Barriers Affecting Higher Education Institutions' Adoption of Blockchain Technology: A Qualitative Study. Informatics, 9(3). DOI: https://doi.org/10.3390/informatics9030064

N, S., Sundaram, B. M., Kumar, V. N., & J, S. a. (2022). Face Recognition based Automated Remote Proctoring Platform. 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), (pp. 1753-1760). Coimbatore, India. DOI: 10.1109/ICAIS53314.2022.9743134.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Retrieved 8 2023, from http://bitcoin.org/bitcoin.pdf

Pee, S. J., Kang, E. S., Song, J. G., & Jang, J. W. (2019). Online test and management system using blockchain network. 2019 21st International Conference on Advanced Communication Technology (ICACT). IEEE. DOI: https://doi.org/10.23919/ICACT.2019.8701891

Rivest, R., Shamir, A., & Adleman, L. (1978, February). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM., 21(2), 120–126. DOI: 10.1145/359340.359342

Sattar, M. R. (2023). An advanced and secure framework for conducting Online examination using blockchain method. Cyber Security and Applications, 1. DOI: https://doi.org/10.1016/j.csa.2022.100005

Swan, M., Raj, P., & Deka, G. C. (2018). Chapter Five - Blockchain for Business: Next-Generation Enterprise Artificial Intelligence Systems. In Advances in Computers (pp. 121-162). Elsevier. DOI: https://doi.org/10.1016/bs.adcom.2018.03.013

Tentea, E.-C., & Ionescu, V. M. (2019). Online quiz implementation using blockchain technology for result tampering prevention. 2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI). IEEE. DOI: 10.1109/ECAI46879.2019.9041980

Tsai, C.-T., Wu, J.-L., Lin, Y.-T., & Yeh, M. K.-C. (2022, July). Design and Development of a Blockchain-Based Secure Scoring Mechanism for Online Learning. Educational Technology & Society, 25(3), 105-121. DOI: https://www.jstor.org/stable/48673728

Viriyasitavat, W., & Hoonsopon, D. (2019). Blockchain characteristics and consensus in modern business processes. Journal of Industrial Information Integration, 32-39. DOI: https://doi.org/10.1016/j.jii.2018.07.004

Zhu, X., & Cao, C. (2021). Secure Online Examination with Biometric Authentication and Blockchain-Based Framework. Mathematical Problems in Engineering. DOI: https://doi.org/10.1155/2021/5058780