INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Smart Contract for Digital Garment Design using Blockchain and Digital Right Management

D Geethanjali[1*], **R Priya**[2], **R Bhavani**[2]

1 Research Scholar, Department of Computer Science and Engineering, Annamalai University, Chidambaram, Tamilnadu, India
2 Professor, Department of Computer Science and Engineering, Annamalai University, Tamilnadu, Chidambaram, India

OPEN ACCESS

Received: 12-01-2022

Accepted: 06-05-2022

Published: 01-07-2022

Citation: Geethanjali D, Priya R, Bhavani R (2022) Smart Contract for Digital Garment Design using Blockchain and Digital Right Management. Indian Journal of Science and Technology 15(24): 1195-1212. https://doi.org/10.17485/IJST/v15i24.84

* **Corresponding author**.

anjali.geetha81@gmail.com

Funding: None

Competing Interests: None

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Background: In the current advancement of communication, anyone can create digital content. Emerging digital marketplaces provides an environment to share digital data with customers who are interested in content, from other digital sources. Digital marketplaces provide a location for both providers and consumers to connect to meet the increased demand. All of this leads to challenges in content protection, copyright protection, contract creation and work trading. **Objectives:** To address these issues, this scheme suggests a blockchain based approach for managing digital rights to garment design works. This system uses the blockchain, digital copyright management techniques and off-chain computation for garment design work. The visibility of design effects, the secrecy of design details and compliance with applicable regulations are all considered. The data delivery is also assured with the help of the proof-of-delivery concept. **Methods:** The proposed system has three steps. They are (i) Creating digital copyright documents using key generation and verification of digital signature (ii) Smart contract creation (iii) Certificate generation for confirmed smart contracts. A Smart contract is defined as computerized transaction protocols that execute the terms of a contract. **Findings:** Smart contracts are created between the designer and customer using Ethereum which is a blockchain based software platform. The interplanetary file system is used to store digital documents. Ethereum blockchain is used to create the smart contract digitally. Ethereum smart contract provides unchangeable, transparent, tamper-proof logs, traceability and responsibility. Finally, E-certificate is generated by the designer for the confirmed contracts and it is uploaded into the IPFS. **Novelty and applications:** The trusted, decentralized and proof of delivery frameworks are included for digital design work with the key features of IPFS, blockchain and Ethereum smart contract. The proposed work is also compared with the existing works based on several criteria such as blockchain, IPFS, PoD, etc.

Keywords: Blockchain; Design work; Copyright Protection; Digital Signature; Ethereum

1 Introduction

Currently, most of the data are stored in a traditional database system. But this solution entails a central authority controlling a vast volume of data, the data's confidentiality, integrity and authenticity cannot be guaranteed. As a result, there are numerous issues in centralized systems, such as Denial of Service(DoS) assaults and single points of failure. This necessitates the use of a distributed system that provides data authenticity, confidentiality and integrity.

At present blockchain is the most widely encouraged technology for securing digital data. Blockchain is being used to tackle inefficiencies in a variety of industries, ranging from identity protection to supply chain management. Blockchain provides the solution for copy-right protection. Several companies are using this technology for attempting to leverage the technology as a tool for managing intellectual property, primarily to register copyrighted works and make it easier for their owners to protect them. It enables photographers to upload photographs and register their works with the copyright office, after which they are time stamped on the blockchain. Creators can then utilize the system to check for infringement on their photographs and use the time-stamped Blockchain to prove ownership. Unfortunately, the copyright owner will still be responsible for enforcing these rights. If the data may be misused by hackers the creators can have the right to take action legally.

IPFS (Interplanetary File System) is a peer-to-peer hypermedia protocol and distributed file system that identifies files through their content. To get file locations and node connectivity information, it uses a Distributed Hash Table (DHT). It has a block storage mechanism and hyperlinks to address the contents. A block denotes a single unit of data, identified by its key or hash. IPFS has no single point of failure because it is distributed.

Smart contracts are simple codes that run when certain criteria are satisfied and are maintained on a blockchain. It is a digital contract or paperless contract which is used to automate the execution of an agreement. It executes without the need of any intermediaries and also saves time. They can also automate a workflow, starting the following step when certain circumstances are satisfied. The smart contract necessity is self-executing contracts in which the contents of the buyer-seller agreement are inscribed directly into lines of code. The Smart contract also extends the transaction's traceability, transparency and irreversibility.

In blockchain, cryptocurrency has been transferred from source to destination. A blockchain transaction is defined as, a state transfer in the blockchain that transforms data from one value to another. A bitcoin transaction, a smart contract, a record, or data storage can all be part of a blockchain transaction. This transaction process is performed, based on following any one of the transaction methods.

On-chain transaction: The public ledger is used to record all the valid transactions performed and it is visible to all the participants on the network. For an on-chain transaction to be complete, there has to be an agreed number of confirmations by miners. The time it takes for an on-chain transaction to complete also depends on network congestion. Therefore, sometimes transactions are delayed if there is a large volume of transactions that need to be confirmed.

Off-chain transaction: It occurs outside of the main blockchain and is not published on the network. The parties concer ned have the option of reaching an agreement outside of the blockchain. It may also include a guarantor, who is responsible for confirming the transaction's completion and certifying that the agreement was followed. The real transaction is then completed on the blockchain after the involved entities, outside the network agree. Different approaches, such as multi-signature technology and credit-based solutions, can be used to carry out these transactions. In comparison to on-chain transactions, off-chain transactions are completed instantly.

The major focus of the proposed system is to exclude the necessity of a third party and also overcome the faith issues between the seller and buyer.

The primary contribution of this paper can be summarized as follows:

• Derive the trusted, decentralized and proof of delivery frameworks are included for digital design work with the key features of IPFS, blockchain and Ethereum smart contract.

• For authentication, the electronic certificate is generated and stored into IPFS for confirmed smart contracts. The designer can download from IPFS whenever they are required.

• Presents smart contract implementation and testing results based on the solidity code.

This paper is organized as follows. Section 2 describes the related works. Section 3 exhibits the key components of the proposed system. Section 4 explains the proposed system. Section 5 provides the implementation and experimental results of the design work. Section 6 presents the comparison between the existing and proposed systems. Section 7 describes the conclusion and future work of the system.

2 Related Work

Josep-Lluis Ferrer-Gomila, et al.[1], presented a multiparty contract signing protocol based on blockchain. Nizamuddina, et al.[2], proposed a digital smart contract based on Ethereum blockchain and IPFS for document version control in the distributed fashion. It automates interactions between various participants, such as developers and approvers. Nishara Nizamuddin, et. al[3], proposed an electronic book sales and purchases. It was implemented using a blockchain framework for the protection of digital resources with the author's royalty. Muqaddas Naz, et.al[4], present the secure sharing of digital data using a blockchain and Interplanetary File System. Keng-Pei Lin, et al.[5], proposed a multi-signature smart contract and blockchain-based mobile ticketing system that protects the tickets' authenticity and security. The system's implementation ensures efficiency at a low cost. Zihao Lu, et al.[6], proposed a new proof of delivery method using smart contracts and digital right management based on blockchain for digital garment design work. It also safeguards trade fairness.

Ahmed W.A.H., et.al[7], proposed a blockchain-enabled supply chain traceability in the textile and apparel sector. Juan José Bullón Pérez,et.al[8], proposed, a transparent supply chain and traceability in the clothing industry which has been implemented using hash codes and digital signatures. Khan, S.N., et.al[9], addressed the analysis of blockchain smart contract applications, current trends and challenges of the corresponding domain. Elva Leka, et.al[10], provides the literature review about the blockchain smart contract and it also analyses the technical challenges of blockchain such as scalability and security. Weidong Fang, et.al[11], proposed an effective digital signature scheme to achieve nonrepudiation. It also investigated and compared in terms of application fields, methods, security, and performance. Omar S. Saleh, et.al[12], proposed a blockchain technology that enhanced certificate verification in the education domain. Amna Qureshi, et.al[13], proposed, reviewed comments and open challenges of blockchain-based multimedia content protection applications. Robert Alexandru Dobre, et.al[14], proposed, blockchain-enabled Image copyright protection using JPEG resistant digital signature. Here the signature should be stored on the blockchain along the identification data of the copyright owner. D.Geethanjali et.al[15], Proposed a blockchain and IPFS based protected copyright data sharing for digital garment design work with digital watermarking and digital right management.

3 Key Components of the proposed system

3.1 Smart Contract

A smart contract is a self-executing, self-enforcing procedure that is directed by its specified terms and conditions and that records and executes contractual agreements using blockchain technology. A Smart contract is usually a self-automated code. Smart contracts have the following benefits. They are speed, efficient and accurate. As a result, no paperwork is required, and no time is lost fixing errors that may occur when filling out documentation by hand. It also ensures that confidence and transparency are maintained. There's no need to worry about information being tampered with for personal gain because there's no third party engaged and encrypted transaction logs are exchanged among participants. (Figure 1) shows the general process flow of the smart contract.

The smart contract is created based on the following steps. Initially, the contract's terms should be determined by the contractual parties. The contractual conditions are then translated into programming code when they have been finalized. Essentially, the code is a collection of conditional statements that explain several circumstances for a future transaction. In the second step, the generated code is stored in blockchain network, where it is replicated among the blockchain members. At last, the code is then run on all machines in the network. When the appropriate condition is met, the contract is immediately

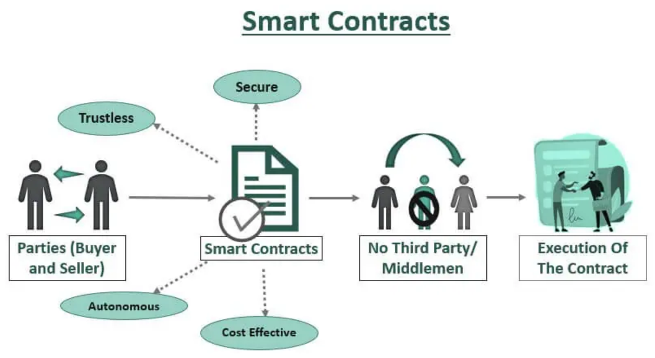

Smart Contracts
Secure
Trustless
Parties (Buyer and Seller)
Smart Contracts
No Third Party/ Middlemen
Execution Of The Contract
Autonomous
Cost Effective

Fig 1. Diagrammatic representation of Smart Contract

executed and the contract created digitally and it is verified by all participants in the blockchain network. So the smart contract is created without the third party.

3.2 Digital Signature

Modern cryptography system has the key, which is used to encrypt the data that can be made public as well as the key that is used to decrypt the data that can be kept private. This cryptography signature provides proof of ownership, tamper-proof, validity and integrity for digital documents. Digital signatures have a vital part in the blockchain. The (Figure 2) shows the process of a typical digital signature.

Like a handwritten signature, each signer's digital signature is unique. Digital signatures use a typical, accepted format, called Public Key Infrastructure (PKI). It also provides the highest level of security and universal acceptance. PKI requires the provider to use a mathematical algorithm to generate two long numbers, called keys. One is a public key and another one is a private key. When a sender electronically signs a document, the signature is created using the sender's private key, which is always securely kept by the sender.

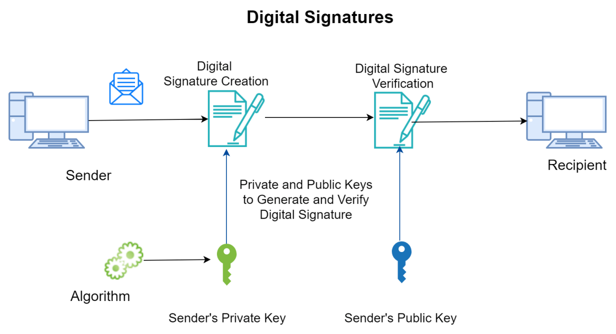

Digital Signatures
Digital Signature Creation
Digital Signature Verification
Sender
Recipient
Private and Public Keys to Generate and Verify Digital Signature
Algorithm
Sender's Private Key
Sender's Public Key

Fig 2. Pictorial representation of Digital Signature creation and verification

The mathematical algorithm acts as a cipher, creating data matching the signed document, called a hash, and encrypting that data. The resulting encrypted data is the digital signature. Both digital data and a copy of the sender's public key are sent to the receiver. After receiving the data, the receiver enters the sender's public key and signature into the algorithm which decrypts digital data. At the time of document signing, the signature is also marked with the time. If the document changes after signing, the digital signature is invalidated.

3.3 Digital Certificate

The registration certificate or electronic certificate is typically used to determine ownership. Blockchain might help with digital rights management by solving the problem of traceable ownership. A claimant must prove ownership of the alleged right as well as its improper appropriation to establish an infringement claim. When the owner is not listed on the registration certificate, however, a claimant must provide a chain of title establishing ownership or exclusive licensee rights. As a result, an author or artist can register a creative work on the time-stamped, immutable blockchain to verify ownership and existence at a specific point in time.

3.4 Elliptic Curve Digital Signature Algorithm (ECDSA

Basically, it is a signature algorithm. Usually, Ethereum and Bitcoin use the Elliptic Curve Digital Signature Algorithm (ECDSA). The major benefit of this algorithm is as follows: Using elliptic curve point manipulation, the user can derive a value from the private key. This key is not reversible, so it is safe and tamperproof. Also, prove the ownership of an address without exposing its private key. It uses less computation power than RSA. It provides robust and efficient encryption. This algorithm ensures that the data or funds can be received by the rightful owner.

3.5 Off-Chain Transaction Process

This smart contract uses the off-chain transaction process. The digital documents are not warehoused on the chain, because it would be prohibitively expensive. But off- chain communication is necessary between the customer and the file server. As a result, an off-chain transaction provides a secure download. The garment designer and the customer are the two key participants in the smart contract. Each entity has an Ethereum address and can participate by invoking smart contract functionalities at specific moments. A straight function call by an entity is not permitted; instead, modifiers are used. The modifiers limit the functions that can be invoked by a few individuals.

3.6 IPFS (Interplanetary File System

IPFS is a distributed file system that aims to connect all computing devices through a single file system. It offers the storing of documents, digital assets, music, videos, photos, and social network information. It also provides the secure storing and retrieval of data with the help of hash values. The IPFS is used to store and retrieved the smart contract documents by using the hash value whenever the user required.

3.7 Fileserver

A file server is a simple computer system with the ability to deliver and receive all file requests over a computer network. The major role of the file is as follows in the proposed system: If all parties agree, the transaction begins, and the agreed-upon collateral is settled from the file server and sent to the customer.

4 Proposed System

The proposed system involves the combination of blockchain technology, Ethereum smart contract and distributed peer-to-peer file system in IPFS. This system provides a decentralized system. This solution also affords the secure distribution of digital data, traceability and reliability. One of the existing smart contract system's shortcomings is using the same work, multiple copyright registrations can be accomplished. To solve this problem, the following two solutions may be used (a) Provide one smart contract per document. (b) The ownership number and document version number are included and uploaded on the chain. It is done based on the designer's request and approval of grant authorities.

The proposed system has the following consecutive steps. They are (i) Copyrighted design documents created using digital signature creation and verification, (ii) smart contract creation (iii) electronic certificate generation for confirmed created contracts. (Figure 3) shows the block diagram of the proposed system.

The proposed system has member participants and they implement the smart contract using the required functionalities.

4.1 Participant Roles in the Smart Contract

The garment design work highlights different participants who are involved and interact with the smart contract. The roles and communications of the members are given as follows:

Garment Designer: The creator of digital design work. A designer can use the blockchain application to register their works for copyright protection. This work's copyright information is validated and the registered work will be sold to the accepted buyer or customer.

Customer: Using the blockchain application, a buyer could undertake a copyright inspection and purchase for registered digital design works, if required.

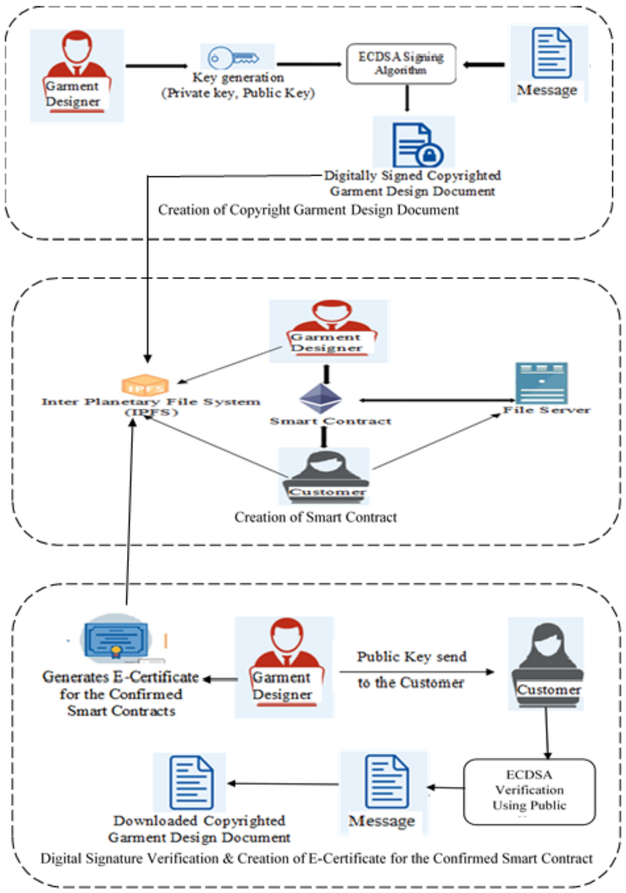

Garment Designer
Key generation (Private key, Public Key)
ECDSA Signing Algorithm
Message
Digitally Signed Copyrighted Garment Design Document
Creation of Copyright Garment Design Document
Garment Designer
Inter Planetary File System (IPFS)
Smart Contract
File Server
Customer
Creation of Smart Contract
Generates E-Certificate for the Confirmed Smart Contracts
Garment Designer
Public Key send to the Customer
Customer
ECDSA Verification Using Public
Message
Downloaded Copyrighted Garment Design Document
Digital Signature Verification & Creation of E-Certificate for the Confirmed Smart Contract

Fig 3. Block diagram of proposed system

File Server : The digital design work data is stored on the file server and can be shared or downloaded by customers. If all parties agree, the transaction begins, and the agreed-upon collateral is settled from the file server and sent to the customer.

4.2 Functionalities of the Smart Contract

Smart contract handles the registration of participants, contract token number generation for customers, copyright file uploading on IPFS, requests for document approval, new registration request and payment settlement for the confirmed contract in the chain. For implementation process establishment the smart contract includes the following to achieve the required functionality:

Methods : Methods are functions that define the contract's functionality. Some methods include limits that only allow a specific entity to use them, while others may be open to all participants. The smart contract's procedures are directly tied to the contract's functionality.

Modifiers : Modifiers can be used to change a function's behavior. They can be used to specify or check a condition prior to execution. Modifiers are used to restrict access to and execution of a smart contract function to only specific participants.

Events : Events are necessary for informing all parties involved about changes to the transactions in progress. An event occurs after the execution of any function call. This also makes it easier to trace back in the event of a disagreement.

Variables : Variables are values that change when conditions or function calls are met. Variables may be able to store certain information depending on the contract.

5 Implementation Steps and Testing Results of the Proposed System

The proposed system is implemented using the Remix IDE. It is used to develop and teffigst the smart contract. It is an open source environment. It can be accessed through the following link https://remix.ethereum.org/. Remix is an environment for building smart contracts using Solidity code, as well as testing and debugging them. This section concentrates on the specifics of implementation and testing. The smart contract is tested in Remix IDE. For testing, the three Ethereum addresses assigned to the participants are as follows:

garment designer: "0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db" customer: "0x5b38da6a701c568545dcfcb03fcb875f56beddc4" and file server: "014723a09acff6d2a60dcdf7aa4aff308fddc160c"

such that initially each of them has 100 Ethers to test the contract code. For deploying the contract, the proposed system uses the JavaScript VM environment in the Remix.

The designer created the garment design work. The design document can be copyrighted either by using watermarking[15] process or by using the ECDSA digital signature algorithm. Using transformation, the design document watermarked into the cover image. This is watermarked image uploaded into the IPFS.

The proposed system is implemented by using the following sequence of steps. They are (i) Copyrighted design document created using digital signature creation and verification (described in section 5.1), (ii) smart contract creation (described in section 5.2) (iii) electronic certificate generation for confirmed created contracts (expressed in section5.3).

5.1 Step1: Process flow of the Digital Signature Generation and Verification

Blockchain technology offers transparency and also provides the capacity to records the each and every log of a transaction on a public ledger. As a result, the Ethereum smart contract generates events and logs that aid in tracking and making proof of delivery easier. The smart contract manages all interactions and transactions among the parties. (Figure 4) shows the process flow of the Creation of Copyright design work using Digital signature and customer's digital signature verification.

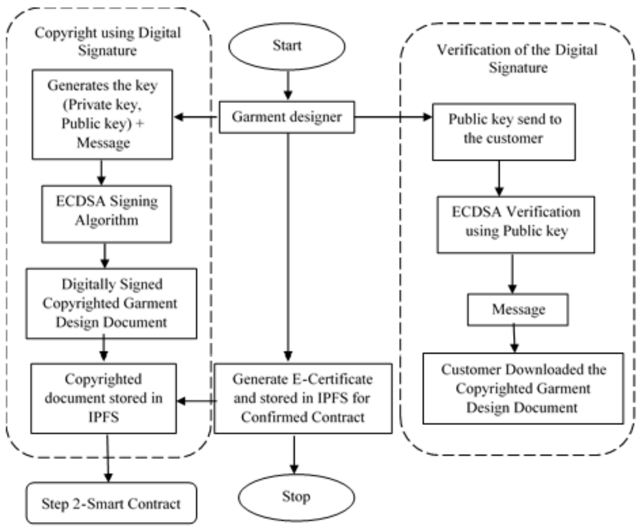

Copyright using Digital Signature
Generates the key (Private key, Public key) + Message
ECDSA Signing Algorithm
Digitally Signed Copyrighted Garment Design Document
Copyrighted document stored in IPFS
Step 2-Smart Contract
Start
Garment designer
Generate E-Certificate and stored in IPFS for Confirmed Contract
Stop
Verification of the Digital Signature
Public key send to the customer
ECDSA Verification using Public key
Message
Customer Downloaded the Copyrighted Garment Design Document

Fig 4. Shows the Copyright data digital signature creation and verification

5.1.1 Digital Signature Generation and Verification

Digital signature is generated based on the following steps (a) Key generation (b) Encryption/Signing (c) Decryption/Verification.

5.1.1.1 Key Creation or generation:. A public and private key will be generated via the key generation. It connects them with the help of an Elliptical Curve Digital Signature Algorithm (ECDSA) and it has the following properties, they are signing and validation. A public key, private key, and message will all be included in a signature. The signature will be returned as another string.

Signature =F (public key, private key, message)

5.1.1.2 Encryption/Signing:.

Encryption is most commonly employed to conceal data within other data. A string or message will be encrypted in the form of hash value. Its goal is to conceal secret message. Signing is used to generate a unique output string, but it also makes the original message public. Algorithm 1 describes the creation of digital signature using ECDSA algorithm.

```
Algorithm 1: Creation of Digital Signature using ECDSA algorithm ECDSA signatures contains of two integers: r and s (where r is 1st half of the bit in signature, s is the second half of the bit in a signature). A variable v is a verification signature or recovery identification used by Ethereum. The signature can be written using the following variables r, s, or v. The sender will require the message to sign and the private key (d⊠) to sign it with in order, to produce a signature. The following is a description of the ”simplified” signature procedure.
1. Hash (e) value calculated from the message(m) to sign.
2. A secure arbitrary value for k is generated.
3. Calculate point (x1, y1) on the elliptic curve by multiplying k with the G constant of the elliptic curve.
4. Calculate r= x1 mod n. If r equals zero, go back to step 2.
5. Compute s= k-1(e+ rda) mod n. If equals zero, go back to step 2.
Where, k is a random number, e is a hash value of message(m), da is private key.
```

5.1.1.3 Decryption/Verification:.

The validation requires only at the receiver’s end to retrieve and verify the data. If the validation function’s output hash value matches the public key, the signature is genuine; otherwise, it’s a forgery.

Validation = F (signature, message)

Case(i) validation=public key which implies the signature and message is valid;

Case (ii) validation $\neq$public key which means the signature and message is not valid.

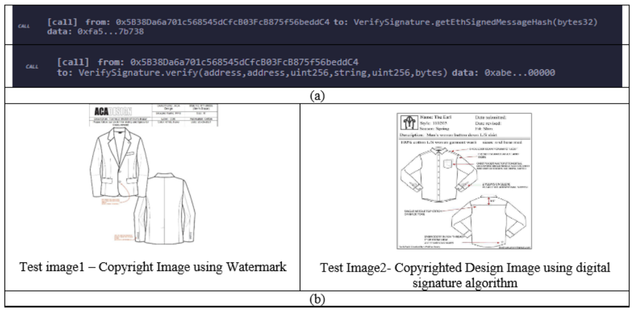
After signing the design document, the designer sends the public key and message to the customer. The customer retrieves the sent document and verifies digital signature using the public key. Algorithm 2 describes the verification algorithm of the ECDSA digital signature.

```
Algorithm 2: Verification algorithm of the ECDSA Digital Signature
1. Hash (e) calculated to recover the message.
2. Compute point R= (x1, y1) on the elliptic curve, where $x_1$ is r for v=27, or r + n for v=28.
3. Determine u1=-zr-1 mod n and u2=sr-1 mod n.
4. Compute point Qa=(xa, ya)=u1G+u2R.
5. Qa = , then reject the signature. Otherwise, convert the x coordinate x1 of Q an integer x11 and calculate v= x11 mod n
6. Accept the signature if and only if v=r.
Where Qa is the point of the public key for the private key used to sign the address. It is used to generate an address and also compare it to the address provided. If it matches the signature is valid. v is the last byte of the signature (v is either 0 or 1). x and y are elliptical coordinates. G is a base point of Elliptical curve. R is the reflection point of the x axis.
```

This environment is only accessible to the authenticated user. The digital signature is decrypted by the customer using the owner’s public key. This assures the authenticity, as only owner has his private key, only the owner can encrypt using his private key, which can then be decrypted using the owner’s public key. For message integrity, the hash value generated by the customer from the message must be the same. If the generated key should be the same every time. If not, the consumer assumes the key is a forgery.

5.1.2 The Copyright document

IPFS is a decentralized file system that produces a unique hash for each file or design document. The registered users only can access and view the garment design document. This specified file can have downloaded and viewed by the authenticated users who know the public key of the document. This copyrighted document includes the information of the garment designer’s id, name, design created date and design specifications of the garment design. The IPFS hash of a test document-1 is “QmZu6JVbsMoyzujjUmJqhQAfDQb8vqvCNiyJ4DEAbc474u” and test document-2 is “QmbEvHM25343kzzrJRuVfTDrZE39QbGYSURgr69Yps5jK4” respectively.



```
CALL  [call]  from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: VerifySignature.getEthSignedMessageHash(bytes32)
      data: 0xfa5...7b738

CALL  [call]  from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
      to: VerifySignature.verify(address,address,uint256,string,uint256,bytes) data: 0xabe...00000
```

(a)
Test image1 – Copyright Image using Watermark
Test Image2- Copyrighted Design Image using digital signature algorithm
(b)

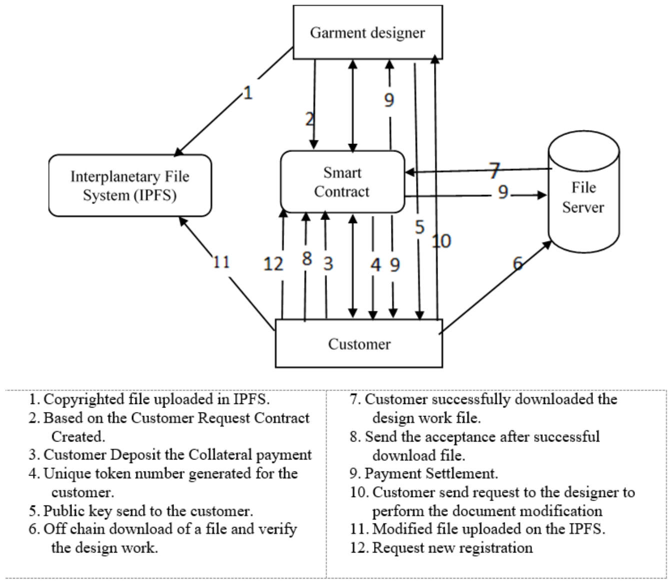
Fig 5. (a) shows the digital signature generation and verification for the design file (b) shows the sample copyrighted garment design document of the designer.

(Figure 5a) shows the message signature generation and verification for the design file. (Figure 5b) shows the sample copyrighted garment design document of the designer.

Encrypted copyright garment design file has been stored in the IPFS by the registered garment designer. This file also contains the terms and conditions of the contract.

5.2 Step 2: Creation of Smart Contract

Figure 6 describes process flow (Step2) and the sequence of activities of a smart contract.



1. Copyrighted file uploaded in IPFS.
2. Based on the Customer Request Contract Created.
3. Customer Deposit the Collateral payment
4. Unique token number generated for the customer.
5. Public key send to the customer.
6. Off chain download of a file and verify the design work.
7. Customer successfully downloaded the design work file.
8. Send the acceptance after successful download file.
9. Payment Settlement.
10. Customer send request to the designer to perform the document modification
11. Modified file uploaded on the IPFS.
12. Request new registration

Fig 6. Shows the process flow and the sequence of activities of a smart contract

The creation of smart contract consists of the following steps:

a. Contract creation request made by the registered customer to designer for agreed digital design document. The registered customers only can access and view the garment design document. If the designer accepts the customers request, the smart contract will be created by the designer.

b. If the contract terms and conditions are accepted by the customer, then the customer deposits the guaranteed amount for confirmation. Then the smart contract automatically generates the contract token number for the specified customers ethereum address.

c. The designer sends the specified file public key to the customer. Now the authenticated customer can download the design document, who knows the encrypted key of the document. After successful smart contract creation, the remaining payment is transferred from the customer to the garment designer and file server.

d. The final copy of the garment design document is uploaded in the IPFS by the designer with the version number. If the contract is not successful, then the guarantee/ collateral amount will be refunded to the customer.

e. If the customer needs to perform any modifications to the existing document, the customer sends the request for designer approval. If any modification is performed in the document, then it will be uploaded to the IPFS with the approval granted by the designer.

f. The new registration request is made by the customer or designer. If the new entry address is not existing, then it will be added to the contract. Otherwise, new registration request will be rejected.

a. **Contract C reation**

Algorithm 3 indicates the process of document request by the customer and based on the customer request the designer creates the contract.

Algorithm 3: Contract Creation and Document Request

```
Input: EA$_c$=Ethereum address of customer, EA$_d$=Ethereum address of designer, S$_{Contract}$ = ContractState
D$_{reg}$ is the Registered Designer
C$_{reg}$ is the Registered Customer
D$_{reg}$ $\leftarrow$C$_{reg}$ Request the Design Document
Restrict access to only
D$_{reg}$$\in$ EA$_d$ D$_{reg}$ $\leftarrow$S$_{Contract}$ is Created
```

The registered customer ethereum address “0x5b38da6a701c568545dcfcb03fcb875f56beddc4” requests the design document. When customers request the design document, it is an automatic payable operation. At the time of customer’s request, 1 ether value is detected from the document requested, customer ethereum address. As per the customer request, the designer creates the smart contract. (Figure 7a) shows the input value, contract execution and event log of the created contract.

The registered garment designer “0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db” makes the contract registration based on the customer request.

b. Contract Token Number generation

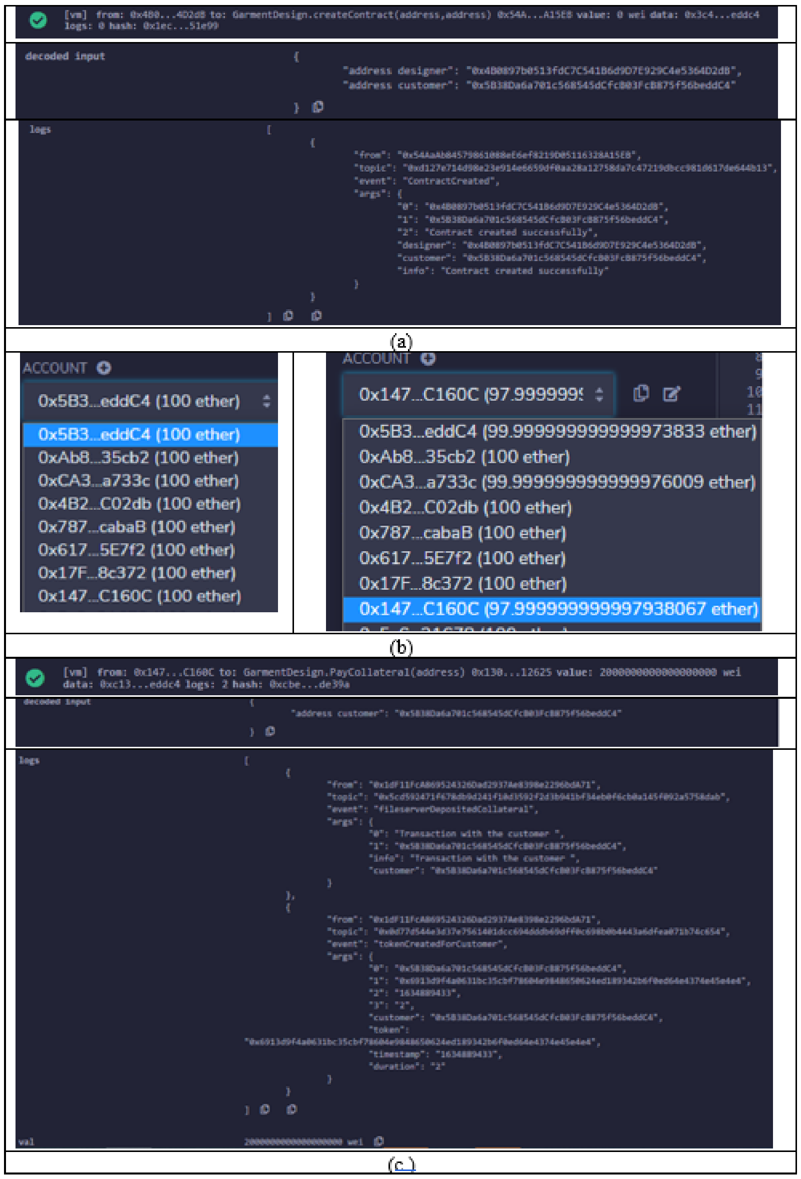
The customer needs to pay the advance payment for contract confirmation. The collateral is 2 Ether because the contract fee is 1 Ether. The file server also paid the collateral of 2 Ethers for the customer and a token was issued for each customer when the customers successfully submitted their payment along with their request. Algorithm 4 describes the sequential steps of the collateral payment.

The customer balance is nearly 99 ethers, which represents the successful contract requisition. The file server balance is 96 ethers, which represents the collateral payment deposited by the file server for the customers. (Figure 7 b) shows the ether balance of all the participants, before and after collateral payment execution of the smart contract.

Algorithm 4 : Collateral Payment

```
Notations Used: EAfs=Ethereum address of file server, EAd=Ethereum address of the designer, EAc=Ethereum address of the customer,
S$_{Contract}$ = ContractState, S$_{Customer}$ =CustomerState, S$_{Designer}$ =DesignerState, TKN$_{Contract}$= Customer Contract Token Number
Input: EAfs, EAd, EAc, IPFS hash, TKN$_{Contract}$, Document Price, Collateral, S$_{Contract}$, S$_{Customer}$ EA= Ethereum Address
S$_{Contract}$ $\leftarrow$ Created
EA Set of all static ethereum addresses saved in this contract Restrict access to any customer C$_{reg}$$\in$ EA
if Contract value=document price +collateral then
S$_{Customer}$ $\leftarrow$ file server Deposited Collateral $\leftarrow$Customer Deposited Collateral
Customer TKN$_{Contract}$ $\leftarrow$ Contract token number generated Successfully
S$_{Customer}$ $\leftarrow$ Received TKN$_{Contract}$ and Hash
else
Revert S$_{Contract}$ and show an error
End
```

After successful the smart contract creation and successful guarantee/collateral payment the contract token number has been generated. This is performed as an internal function of smart contract. The contract token number is built using a keccak256 hash. It is an in-built hash process which is especially created by the ethereum blockchain. This hash is generated by using the TokenGeneration () function and which has the following data, the Ethereum address of the customer (EA$_{cus}$), number of customers (NoC), number of successful sales (NoS), Token validity (TV), and time stamp of the block(Block$_{Timestamp}$). It can be written as Token$_{Customer}$= keccak256(EA$_{cus}$, NoC, NoS, TV, Block$_{Timestamp}$). This contract token number creation is providing



```
[vm] from: 0x4B0...4D2dB to: GarmentDesign.createContract(address,address) 0x54A...A15E8 value: 0 wei data: 0x3c4...eddc4
logs: 0 hash: 0x1ec...51e99

decoded input                     {
                                          "address designer": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                                          "address customer": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                                  }

logs                           [
                                    {
                                          "from": "0x54AaAb84579861088eE6ef8219D0511632BA15E8",
                                          "topic": "0xd127e714d98e23e914e6659df0aa28a12758da7c47219dbcc981d617de644b13",
                                          "event": "ContractCreated",
                                          "args": {
                                                "0": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                                                "1": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                "2": "Contract created successfully",
                                                "designer": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                                                "customer": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                "info": "Contract created successfully"
                                          }
                                    }
                               ]
```

(a)

```
ACCOUNT
0x5B3...eddC4 (100 ether)
0x5B3...eddC4 (100 ether)
0xAb8...35cb2 (100 ether)
0xCA3...a733c (100 ether)
0x4B2...C02db (100 ether)
0x787...cabaB (100 ether)
0x617...5E7f2 (100 ether)
0x17F...8c372 (100 ether)
0x147...C160C (100 ether)
```

```
ACCOUNT
0x147...C160C (97.9999999
0x5B3...eddC4 (99.999999999999973833 ether)
0xAb8...35cb2 (100 ether)
0xCA3...a733c (99.999999999999976009 ether)
0x4B2...C02db (100 ether)
0x787...cabaB (100 ether)
0x617...5E7f2 (100 ether)
0x17F...8c372 (100 ether)
0x147...C160C (97.999999999997938067 ether)
```

(b)

```
[vm] from: 0x147...C160C to: GarmentDesign.PayCollateral(address) 0x130...12625 value: 2000000000000000000 wei
data: 0xc13...eddc4 logs: 2 hash: 0xcbe...de39a

decoded input                  {
                                     "address customer": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                               }

logs                        [
                                 {
                                       "from": "0x1dF11fcA8695243260ad29374e8398e2296bdA71",
                                       "topic": "0x5cd592471f678db9d241f10d3592f2d3b941bf34eb0f6cb0a145f092a5758dab",
                                       "event": "fileserverDepositedCollateral",
                                       "args": {
                                             "0": "Transaction with the customer ",
                                             "1": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                             "info": "Transaction with the customer ",
                                             "customer": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                                       }
                                 },
                                 {
                                       "from": "0x1dF11fcA8695243260ad29374e8398e2296bdA71",
                                       "topic": "0x0d77d544e3d37e7561401dcc694dddb69dff0c698b0b4443a6dfea071b74c654",
                                       "event": "tokenCreatedForCustomer",
                                       "args": {
                                             "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                             "1": "0x6913d9f4a0631bc35cbf78604e98486506 24ed189342b6f0ed64e4374e45e4e4",
                                             "2": "1634889433",
                                             "3": "2",
                                             "customer": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                             "token":
                            "0x6913d9f4a0631bc35cbf78604e9848650624ed189342b6f0ed64e4374e45e4e4",
                                             "timestamp": "1634889433",
                                             "duration": "2"
                                       }
                                 }
                            ]

val                         2000000000000000000 wei
```

(c)

Fig 7. (a) Input value, Contract execution message and event log of the created contract (b) Shows the ether balance of all the participants, before and after collateral payment execution of the smart contract (c) shows the Collateral Payment execution and event log for the created customer token number of the smart contract.

the a unique identity to each customer. (Fig.7c) shows the collateral payment execution and event log of the token generation of the customer with the ethereum address “0x5b38da6a701c568545dcfcb03fcb875f56beddc4”.

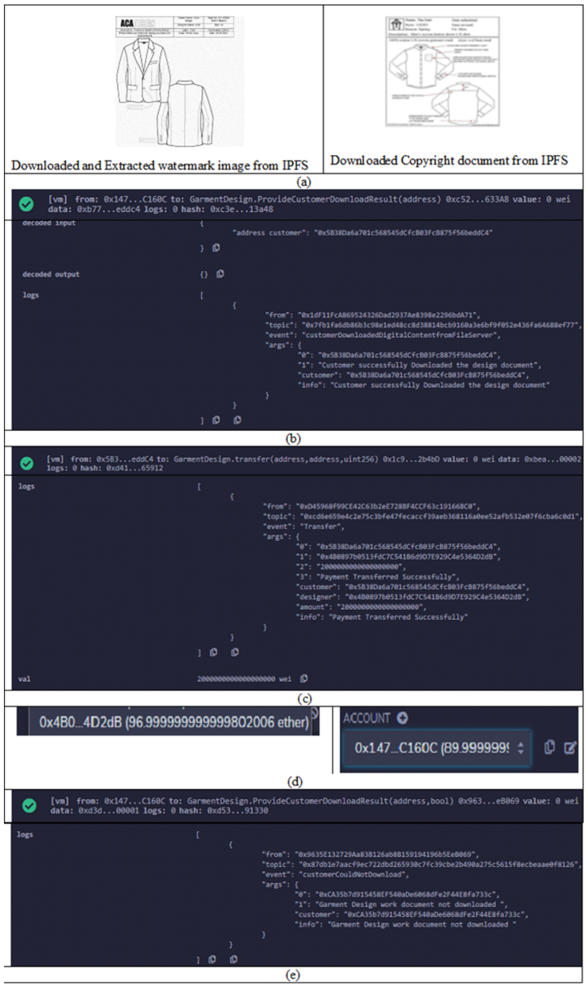
c. Successful Transaction

Algorithm 5 represents the consecutive steps of the design document download, successful and unsuccessful payment settlement.

The customer ethereum address, contract token number and design document hash data are checked by the designer. If the data are correct then the designer sends the specified file public key to the customer. Now the authenticated customer can download the design document, who knows the encrypted key of the document. (Figure 8 a) shows the customer sample downloaded file from IPFS.

Algorithm 5: Document download and payment Settlement

```
Input: EAfs, EAd, EAc, $S_{Contract}$ , $S_{Customer}$ , $S_{designer}$
Ethereum address of fileserver, customer and designer address saved in the contract.
if $S_{Customer}$ ==Successful Download and satisfied then
Contract remaining value to filesever fspay= collateral + x Contract remaining value to designer despay= docprice/2 EAfs$\leftarrow$fspay$\leftarrow$EAc
//Customer pay the remaining contract payment to fileserver
EAd$\leftarrow$despay$\leftarrow$EAc //Customer pay the remaining contract payment to designer Customer successfully settle the payment $S_{Customer}$
$\leftarrow$Transaction Completed
$S_{Designer}$$\leftarrow$Generate E certificate for the Successful contract
else
$S_{Customer}$==Unsuccessful Download or not satisfied then
Check the Customer $TKN_{Contract}$
Fileserver$\rightarrow$Refund the Collateral to Customer
$S_{Customer}$ $\leftarrow$Payment Refunded Successfully
else
Revert $S_{Contract}$ and show an error
End
```


Downloaded and Extracted watermark image from IPFS
Downloaded Copyright document from IPFS
(a)

```
[vm] from: 0x147...C160C to: GarmentDesign.ProvideCustomerDownloadResult(address) 0xc52...633A8 value: 0 wei
data: 0xb77...eddc4 logs: 0 hash: 0xc3e...13a48
decoded input                       {
                                          "address customer": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                                    }
decoded output                      {}
logs                                [
                                          {
                                                "from": "0x1dF11FcA869524326Dad2937Ae8398e2296bdA71",
                                                "topic": "0x7fb1fa6db86b3c98e1ed48cc8d38814bcb9160a3e6bf9f052e436fa64688ef77",
                                                "event": "customerDownloadedDigitalContentfromFileServer",
                                                "args": {
                                                      "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                      "1": "Customer successfully Downloaded the design document",
                                                      "cutsomer": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                      "info": "Customer successfully Downloaded the design document"
                                                }
                                          }
                                    ]
```

(b)

```
[vm] from: 0x5B3...eddC4 to: GarmentDesign.transfer(address,address,uint256) 0x1c9...2b4bD value: 0 wei data: 0xbea...00002
logs: 0 hash: 0xd41...65912
logs                                [
                                          {
                                                "from": "0xD45968f99CE42C63b2eE728BF4CCF63c19166BC0",
                                                "topic": "0xcd6e659e4c2e75c3bfe47fecaccf39aeb368116a0ee52afb532e07f6cba6c0d1",
                                                "event": "Transfer",
                                                "args": {
                                                      "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                      "1": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                                                      "2": "2000000000000000000",
                                                      "3": "Payment Transferred Successfully",
                                                      "customer": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                      "designer": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                                                      "amount": "2000000000000000000",
                                                      "info": "Payment Transferred Successfully"
                                                }
                                          }
                                    ]
val                                 2000000000000000000 wei
```

(c)

```
0x4B0...4D2dB (96.999999999999802006 ether)
ACCOUNT
0x147...C160C (89.999999
```

(d)

```
[vm] from: 0x147...C160C to: GarmentDesign.ProvideCustomerDownloadResult(address,bool) 0x963...eB069 value: 0 wei
data: 0xd3d...00001 logs: 0 hash: 0xd53...91330
logs                                [
                                          {
                                                "from": "0x9635E132729Aa838126ab8B159194196b5Ee8069",
                                                "topic": "0x87db1e7aacf9ec722dbd265930c7fc39cbe2b490a275c5615f8ecbeaae0f8126",
                                                "event": "customerCouldNotDownload",
                                                "args": {
                                                      "0": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c",
                                                      "1": "Garment Design work document not downloaded ",
                                                      "customer": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c",
                                                      "info": "Garment Design work document not downloaded "
                                                }
                                          }
                                    ]
```

(e)

Fig 8. (a) shows the customer sample downloaded file from IPFS (b) shows the successful document download result and event log of the customer in the smart contract(c) shows the Ether payment settlement by the customer to the designer for the smart contract (d) shows the Ether balance of all the participants for the successful payment settlement of the smart contract (e) shows the unsuccessful transaction and event log of the contract disagreement operation

In the successful transaction, the customer downloaded the design document effectively which is indicated by the smart contract file server. (Figure 8b) shows the successful document download result and event log of the customer in the smart contract.

If the customer is satisfied with the downloaded document, the remaining contract payment is settled to the designer as per the contract guidelines. If the total contract amount is 6 ethers mean, already the collateral payment is 2 ethers paid by the customer. The remaining 4 ethers need to be paid by the customer. This 4 ethers amount is shared by the file server and the designer, 2 ethers each respectively. (Figure 8c) shows the Ether payment settlement by the customer to the designer for the smart contract.

The customer ethereum address "0x5b38da6a701c568545dcfcb03fcb875f56beddc4" paid the remaining 2 ethers to the designer "0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db". As like customer transfer the remaining 2 ethers to the fileserver "0x14723a09acff6d2a60dcdf7aa4aff308fddc160c". (Figure 8d) shows the ether balance of all the participants for the successful payment settlement for the smart contract.

After successful payment, the designer uploads the final copy of the design document in the IPFS with the version number for further reference.

d. Unsuccessful Transaction

If the customer disagrees with the contract the collateral amount must be refunded to the customer. The designer checks the token number of the customer. The designer downloads the disagreed smart contract information using off-chain. The collateral payment settled as like successful transaction. The collateral amount of 4 ethers should be refunded, 2 ethers for the customer and 2 ethers for the fileserver. The ethereum address "0xca35b7d915458ef540ade6068dfe2f44e8fa733c" given as an input parameter. (Figure 8e) shows the unsuccessful transaction since the file cannot be downloaded effectively.

e. Document Modification/Uploading Approval

If the customer required any modification in the contract design document or terms and conditions, the customer must get the approval from the designer. The registered customer requests the designer for document modification. Based on the request, the designer checks the customer address and document hash. Algorithm 6 represents the steps of the design document modification request by the customer.

Algorithm 6: Design Document Modification Request

```
Input: EAc, EAd, IPFS Hash, $C_{reg}$, $D_{reg}$, $S_{Contract}$ ,$S_{Customer}$,$S_{Designer}$
$S_{Contract}$==Created
$S_{Customer}$ is ReadytoSubmit
Restrict access to only $D_{reg} \in$ EAd
if $C_{reg}$ (EAc) and IPFS hash=true then
$S_{Designer}$ ==AcceptModificationRequest
$S_{Contract}$ $\leftarrow$WaitForDesignersApproval
$S_{Customer}$ $\leftarrow$SubmittedForApproval
$S_{Contract}$ $\rightarrow$ GrantApproval for document modification
else
$C_{reg}$ (EAc) and IPFS hash$\neq$true then
$S_{Designer}$ $\neq$AcceptModificationRequest
$S_{Contract}$ $\leftarrow$WaitForDesignersApproval
$S_{Customer}$ $\leftarrow$SubmittedForApproval
$S_{Contract}$ $\leftarrow$ApprovalRejected for document modification
else
Revert $S_{Contract}$ and show an error
End
```

Case (i): Customer Address, Document Hash == Matched with existing smart contract =>Approval granted to document modification (if designer accepts the modification)

If Customer address and document hash matches with smart contract data, then the state of the request changed from Waitfordesignerapproval state to SubmittedforApproval state. The customer gives a request with the input parameters such as customer's ethereum address "0x5b38da6a701c568545dcfcb03fcb875f56beddc4" and the IPFS hash of a garment design document "QmbEvHM25343kzzrJRuVfTDrZE39QbGYSURgr69Yps5jK4". Based on the given inputs requestforApproval function executes and the event log shows the approval status of the customer. (Figure 9 a) displays the execution of the document modification approval request function and corresponding event log of the smart contract.

Algorithm 7: Uploading the Modified design Document

```
Input: EAc , $S_{Contract}$, $S_{Customer}$, $S_{Designer}$
$S_{Contract}$ ←Wait For Designers Approval
$S_{Customer}$ ← Ready to Submit S
$_{Designer}$ ← Waiting To Sign
Restrict access to only $D_{reg} \in EA_d$
if document hash[$EA_{Cus}$]=IPFSHash of Document then
$S_{Contract}$ ←Approval Provided
$S_{Customer}$ ←Approval Granted
$S_{Designer}$ ←Approval Success
Customer uploads the modified Document with the version number
else
$S_{Contract}$ ←Approval Denied
$S_{Customer}$ ← Approval Not Provided
$S_{Designer}$ ←Approval Failed
Customer not permitted to perform the document modification
else
Revert $S_{Contract}$ and show error
End
```

Algorithm 7 represents the uploading the modified design document based on the customer request.

The designer provides the approval to modify the document, then the customer's performs the granted modifications and it sends to the designer for approval. The designer verifies the modified data. If the modification is accepted by the designer, then the transaction state of the customer changed from NewVersionSigned to Approval Granted while executing the provide Approval To modified and Upload () function. The successful upload approval grant, will be permitted by the designer. This modified design document is uploaded to the IPFS with the hash and the version number of the document. The customer provides their ethereum address as an input "0x5b38da6a701c568545dcfcb03fcb875f56beddc4" to execute the provide Approval To modified and Upload () function. (Figure 9b) displays the successful execution and event log of modified document upload approval of the smart contract.

Case (ii): Customer Address, Document Hash $\neq$ Matched with existing smart contract =>Approval not Granted to Document modification (if designer not accepts the modification)

If the input value (customer address or document hash) not matched with the existing contract data, the designer will not accept the customer modification request. Then the designer will not provide the ApprovalGrant for document modification. Here, the contract transaction state reformed from SignatureDenied to ApprovalFailed. The customer gets the Approval Rejected information from the designer. So it is represented as failed transaction or approval rejected for document modification. (Figure 9c) displays the event log of the rejected approval for document modification in the smart contract.

Case (iii): If the function executed by unregistered entity.

If any of the values are (customer address or document hash) is not matched with the existing contract data, it implies the customer request condition is failed, so the transaction is terminated and it revert back to the initial state. In this case, the customer is not considered as a registered member of the contract. So the designer will not provide the permission for modification. So it is represented as a failed transaction. (Figure 9 d) displays the error message of the unsuccessful execution and event log of the document modification rejection of the unregistered member in the smart contract.

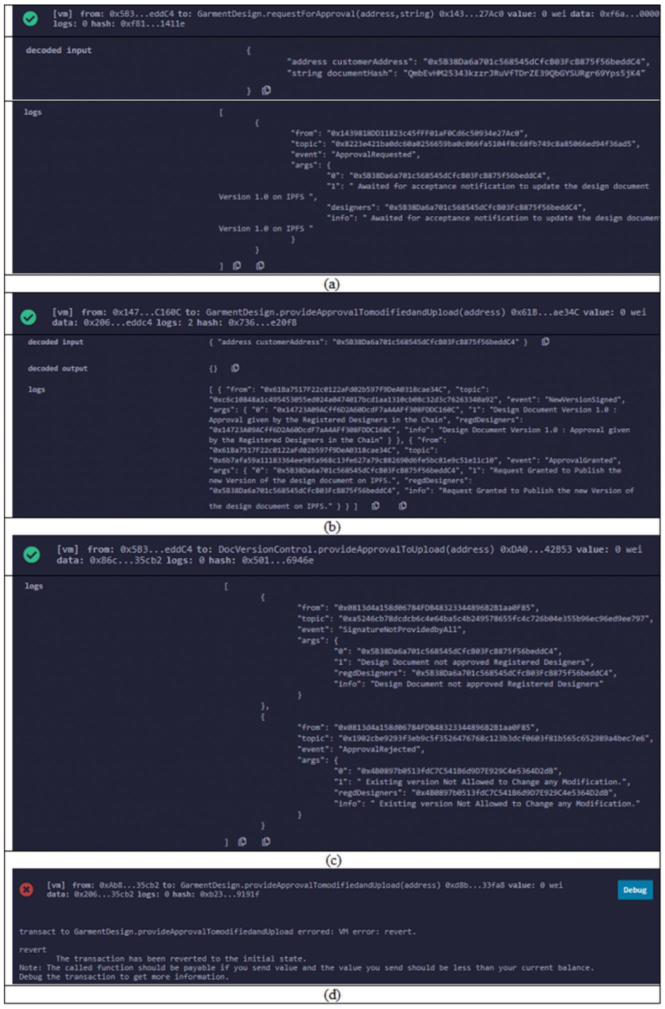
f. Request new registration entry address/Approval

This section grants the new registration entry as per the designer or customer request. The designer or customer provides the ethereum address as an input value. This contract checks the new registration request. Algorithm 8 describes the new registration request of the unregistered member.

Case (i): If new entry address $\neq$ Existing address=> Approval granted

The new entry address given by the designer or customer. The requesting ethereum address state is checked in the contract whether the address is existing or not. If the address is not available, then the new address transaction state is modified from WaitToRegister state to NewRegistrtaionRequested. The successful new registration request, will receive a grant approval for new registration.

New registration request by the customer or designer by executing the requestNewRegistration() function. It is tested by a following new entry ethereum address as an input "0xca35b7d915458ef540ade6068dfe2f44e8fa733c". It is a new address, so it receives the registration approval from the smart contract. (Figure 10 a) shows the successful execution of the



```
[vm] from: 0x5B3...eddC4 to: GarmentDesign.requestForApproval(address,string) 0x143...27Ac0 value: 0 wei data: 0xf6a...0000
logs: 0 hash: 0xf81...1411e

decoded input                  {
                                    "address customerAddress": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                    "string documentHash": "QmbEvHM25343kzzrJRuVFTDrZE39QbGYSURgr69Yps5jK4"
                               }

logs                       [
                                {
                                    "from": "0x1439818DD11823c45fFF01aF0Cd6c50934e27Ac0",
                                    "topic": "0x8223e421ba0dc60a0256659ba0c066fa5104f8c68fb749c8a85066ed94f36ad5",
                                    "event": "ApprovalRequested",
                                    "args": {
                                         "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                         "1": " Awaited for acceptance notification to update the design document
                           Version 1.0 on IPFS ",
                                         "designers": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                         "info": " Awaited for acceptance notification to update the design documen
                           Version 1.0 on IPFS "
                                    }
                                }
                           ]
```

(a)

```
[vm] from: 0x147...C160C to: GarmentDesign.provideApprovalTomodifiedandUpload(address) 0x61B...ae34C value: 0 wei
data: 0x206...eddc4 logs: 2 hash: 0x736...e20f8

decoded input              { "address customerAddress": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4" }

decoded output             {}

logs                       [ { "from": "0x61Ba7517F22c0122aFd02b597f9DeA0318cae34C", "topic":
                           "0xc6c10848a1c495453055ed024a0474017bcd1aa1310cb08c32d3c76263340a92", "event": "NewVersionSigned",
                           "args": { "0": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C", "1": "Design Document Version 1.0 :
                           Approval given by the Registered Designers in the Chain", "regdDesigners":
                           "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C", "info": "Design Document Version 1.0 : Approval given
                           by the Registered Designers in the Chain" } }, { "from":
                           "0x61Ba7517F22c0122aFd02b597f9DeA0318cae34C", "topic":
                           "0x6b7afa59a11183364ee985a968c13fe627a79c882690d6fe5bc81e9c51e11c10", "event": "ApprovalGranted",
                           "args": { "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4", "1": "Request Granted to Publish the
                           new Version of the design document on IPFS.", "regdDesigners":
                           "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4", "info": "Request Granted to Publish the new Version of
                           the design document on IPFS." } } ]
```

(b)

```
[vm] from: 0x5B3...eddC4 to: DocVersionControl.provideApprovalToUpload(address) 0xDA0...42B53 value: 0 wei
data: 0x86c...35cb2 logs: 0 hash: 0x501...6946e

logs                       [
                                {
                                    "from": "0x0813d4a158d06784FDB48323344896B2B1aa0F85",
                                    "topic": "0xa5246cb78dcdcb6c4e64ba5c4b249578655fc4c726b04e355b96ec96ed9ee797",
                                    "event": "SignatureNotProvidedbyAll",
                                    "args": {
                                         "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                         "1": "Design Document not approved Registered Designers",
                                         "regdDesigners": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                         "info": "Design Document not approved Registered Designers"
                                    }
                                },
                                {
                                    "from": "0x0813d4a158d06784FDB48323344896B2B1aa0F85",
                                    "topic": "0x1902cbe9293f3eb9c5f3526476768c123b3dcf0603f81b565c652989a4bec7e6",
                                    "event": "ApprovalRejected",
                                    "args": {
                                         "0": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                                         "1": " Existing version Not Allowed to Change any Modification.",
                                         "regdDesigners": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                                         "info": " Existing version Not Allowed to Change any Modification."
                                    }
                                }
                           ]
```

(c)

```
[vm] from: 0xAb8...35cb2 to: GarmentDesign.provideApprovalTomodifiedandUpload(address) 0xd8b...33fa8 value: 0 wei
data: 0x206...35cb2 logs: 0 hash: 0xb23...9191f

transact to GarmentDesign.provideApprovalTomodifiedandUpload errored: VM error: revert.

revert
        The transaction has been reverted to the initial state.
Note: The called function should be payable if you send value and the value you send should be less than your current balance.
Debug the transaction to get more information.
```

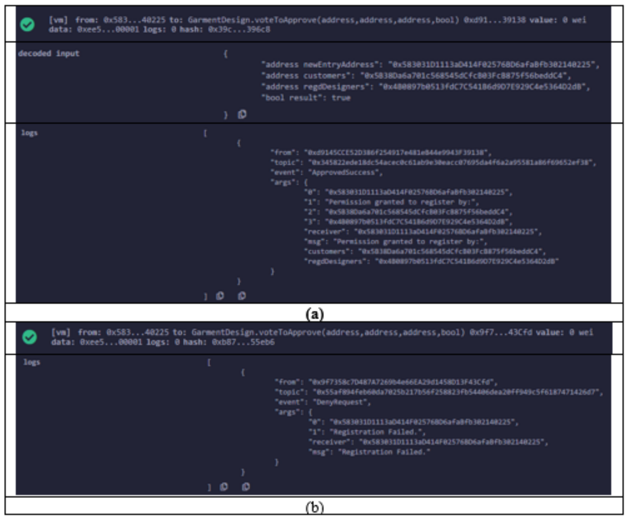
(d)

Fig 9. (a) displays the execution and event log of document modification approval request function of the smart contract (b) displays the successful execution and event log of modified document upload approval of the smart contract. (c) displays the event log of the rejected approval for document modification in the smart contract (d) displays the error message of the provide Approval To modified and upload() function executed by unregistered member in the smart contract.

Algorithm 8: New Registration Request of the unregistered member

Input: EA_{New}= Ethereum address new registration requested

```
$S_{Contract}$ is Approval Provided
New Registration State is Wait to Register
if $EA_{new}$ is ==Exist then
Contract State reverts and shows an error
else
Contract state changes to $EA_{New}$
New RegistrationState $\leftarrow$ $EA_{New}$
$C_{Designer}$ changes to ApprovalFailed New Registration Request $\leftarrow$ Permission Granted
else
Revert $S_{Contract}$ and show an error
End
```



```
[vm] from: 0x583...40225 to: GarmentDesign.voteToApprove(address,address,address,bool) 0xd91...39138 value: 0 wei
data: 0xee5...00001 logs: 0 hash: 0x39c...396c8

decoded input                    {
                                        "address newEntryAddress": "0x583031D1113aD414F025768D6afaBfb302140225",
                                        "address customers": "0x5838Da6a701c568545dCfcB03Fc8875f56beddC4",
                                        "address regdDesigners": "0x480897b0513fdC7C54186d9D7E929C4e5364D2dB",
                                        "bool result": true
                                 }

logs                          [
                                    {
                                          "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
                                          "topic": "0x345822ede18dc54acec0c61ab9e30eacc07695da4f6a2a95581a86f69652ef38",
                                          "event": "ApprovedSuccess",
                                          "args": {
                                                "0": "0x583031D1113aD414F025768D6afaBfb302140225",
                                                "1": "Permission granted to register by:",
                                                "2": "0x5838Da6a701c568545dCfcB03Fc8875f56beddC4",
                                                "3": "0x480897b0513fdC7C54186d9D7E929C4e5364D2dB",
                                                "receiver": "0x583031D1113aD414F025768D6afaBfb302140225",
                                                "msg": "Permission granted to register by:",
                                                "customers": "0x5838Da6a701c568545dCfcB03Fc8875f56beddC4",
                                                "regdDesigners": "0x480897b0513fdC7C54186d9D7E929C4e5364D2dB"
                                          }
                                    }
                              ]
```

(a)

```
[vm] from: 0x583...40225 to: GarmentDesign.voteToApprove(address,address,address,bool) 0x9f7...43Cfd value: 0 wei
data: 0xee5...00001 logs: 0 hash: 0xb87...55eb6

logs                          [
                                    {
                                          "from": "0x9f7358c7D487A7269b4e66EA29d1458D13F43Cfd",
                                          "topic": "0x55af894feb60da7025b217b56f258823fb54406dea20ff949c5f6187471426d7",
                                          "event": "DenyRequest",
                                          "args": {
                                                "0": "0x583031D1113aD414F025768D6afaBfb302140225",
                                                "1": "Registration Failed.",
                                                "receiver": "0x583031D1113aD414F025768D6afaBfb302140225",
                                                "msg": "Registration Failed."
                                          }
                                    }
                              ]
```

(b)

Fig 10. (a) Shows the successful execution of the request New Registration () function and the corresponding event log of new registration approval grant in smart contract (b) Shows the registration failed message of the already registered user in the smart contract.

requestNewRegistration() function and the corresponding event log of new registration approval grant in smart contract.

Case (ii): If new entry address == already registered =>New Registration Failed

New registration request by the customer or designer by executing the requestNewRegistration() function. If the ethereum address is already registered, then it displays the error message. The following ethereum address is given as an input "0x583031d1113ad414f02576bd6afabfb302140225". It has been already registered. (Figure 10 b) shows the registration failed message of the already registered user in the smart contract.

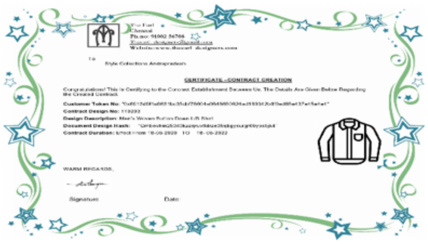
Step 3: Generating E-certificate for confirmed Smart contracts

a. For a successful smart contract, the e-certificate is generated by the designer and uploaded to the IPFS.

b. The customer can download the certificate for their authentication.

Generating E-certificate for Confirmed Smart Contracts

The designer checks the status of the payment settlement for the confirmed smart contracts. If the payment settlement is finished, then the designer will create the e-certificate for this successful smart contract. The e-certificate contains the details of customer token no, customer id, IPFS Hash of a file, and a description about the design work and duration of the contract. This e-certificate is uploaded to the IPFS file system. For authentication the customer can download the e-certificate whenever they require. (Figure 10 c shows the e-certificate of the confirmed smart contract for the garment design work.



[illegible]
Chennai
Ph.no: 91002 56786
[illegible]
Website:www.[illegible] designers.com

To
Style Collections Andrapradesh

CERTIFICATE –CONTRACT CREATION

Congratulations! This Is Certifying to the Contract Establishment Between Us. The Details Are Given Below Regarding the Created Contract.

Customer Token No: [illegible]
Contract Design No: 113232
Design Description: Men's Woven Button Down L/S Shirt
Document Design Hash: [illegible]
Contract Duration: Effect From 16-06-2020 TO 16-06-2022

WARM REGARDS,

[illegible]

Signature Date

Fig 11. e-certificate of the confirmed smart contract for the garment designwork

6 Comparison of Existing Work

The proposed work compared with the following common terms used in the References: Blockchain, Data encryption, Interplanetary File System(IPFS), Proof-of-Delivery (PoD), Digital Right Management (DRM), off-chain transaction process and digital watermarking(DW). The comparison of existing work with the proposed framework is given in Table 1.

Table 1. Comparison between the existing and proposed work

State of Art	Application Described	**Block chain**	**Data Encryption**	**IPFS**	**PoD**	**DRM**	**Off-chain**	**DW**
Reference [2]	Document version control of the digital data	yes	No	yes	no	no	yes	No
Reference [3]	Author Royalty	yes	No	yes	yes	yes	yes	No
Reference [4]	Sharing digital data	yes	yes	yes	yes	no	no	No
Reference [6]	Design work	yes	yes	no	yes	yes	yes	No
Reference [8]	Research data right management	Yes	No	no	no	yes	no	No
Reference [9]	Sale of Digital Content	Yes	No	yes	yes	no	yes	No
Reference [12]	Review based system	Yes	No	no	no	no	no	No
Proposed system	Digital garment design work	Yes	yes	yes	yes	yes	yes	Yes

7 Conclusion and Future Work

This paper, presented a blockchain based solution and framework for digital garment design work with proof of delivery. Digital copyrighted documents are created and verified by using the digital signature algorithm. The copyrighted digital signature authentication or watermarked data provides the secure sharing of data on the network. Blockchain and smart contracts can improve the owner's identity and security. The proposed system was implemented and tested by using the RemixIDE for the following functionalities such as design document copyrights, document versions and payment settlements. The blockchain and IPFS decentralized file system ensures the secure storing of design data, good productivity and it also maintains the smart contract agreement between the parties with reliability. The ethereum network, illustrates the blockchain-based platform that enables the off-chain transaction process. The proposed work compared with the existing work based on several criteria such as blockchain, IPFS etc. The future work of the system can be extended with multi party smart contract, traceability with QR (Quick Response) code and analysis of various attacks of the smart contract.

References

1) Ferrer-Gomila JL, Hinarejos MF. A Multi-Party Contract Signing Solution Based on Blockchain. *Electronics*. 2021;10(12):1457–1457. Available from: https://doi.org/10.3390/electronics10121457.
2) Nizamuddina N, Salaha K, Azadb M, Arshadc J, Rehman MH. Decentralized document version control using ethereum blockchain and IPFS. *Computers and Electrical Engineering*. 2019;76:183–197. Available from: https://doi.org/10.1016/j.compeleceng.2019.03.0140045-7906/.
3) Nizamuddin N, Hasan H, Salah K, Iqbal R. Blockchain-Based Framework for Protecting Author Royalty of Digital Assets. *Arabian Journal for Science and Engineering*. 2019;44(4):3849–3866. Available from: https://doi.org/10.1007/s13369-018-03715-4.
4) Naz M, Fahad A, Al-Zahrani R, Khalid N, Javaid AM, Qamar MK, et al. A Secure Data Sharing Platform Using Blockchain and Interplanetary File System. *Sustainability* . 2019;11:7054–7054. Available from: https://doi.org/10.3390/su11247054.
5) Lin KP, Chang YW, Wei ZH, Shen CY, Chang MY. A Smart Contract-Based Mobile Ticketing System with Multi-Signature and Blockchain. *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*. 2019;p. 231–232.
6) Lu Z, Shi Y, Tao R, Zhang Z. Blockchain for Digital Rights Management of Design Works. *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. 2019.
7) Ahmed WAH, Maccarthy BL. Blockchain-Enabled Supply Chain Traceability in the Textile and Apparel Supply Chain: A Case Study of the Fiber Producer, Lenzing. *Sustainability*. 2021;13(19):10496–10496. Available from: https://doi.org/10.3390/su131910496.
8) Pérez JJB, Queiruga-Dios A, Víctor G, Ángel Martín Del Martínez, Rey. Traceability of Ready-to-Wear Clothing through Blockchain Technology. *Sustainability*;2020(18):7491–7491. Available from: https://doi.org/10.3390/su12187491.
9) Khan SN, Loukil F, Ghedira-Guegan C, Benkhelifa E, Bani-Hani A. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-Peer Networking and Applications*. 2021;14(5):2901–2925. Available from: https://doi.org/10.1007/s12083-021-01127-0.
10) Leka E, Selimi B, Lamani L. Systematic Literature Review of Blockchain Applications: Smart Contracts. In: 2019 International Conference on Information Technologies (InfoTech). IEEE. 2019. Available from: https://doi.org/10.1109/infotech.2019.8860872.
11) Fang W, Chen W, Zhang W, Pei J, Gao W, Wang G. Digital signature scheme for information non-repudiation in blockchain: a state of the art review. *EURASIP Journal on Wireless Communications and Networking*. 2020;2020(1). Available from: https://doi.org/10.1186/s13638-020-01665-w.

12) Saleh OS, Ghazali O, Ehsan RM. Blockchain based Framework for Educational Certificates Verification. *Journal of Critical Reviews*. 2020;7.
13) Qureshi A, Jiménez DM. Blockchain-Based Multimedia Content Protection: Review and Open Challenges. *Applied Sciences*;11(1):1–1. Available from: https://doi.org/10.3390/app11010001.
14) Dobre RA, Preda RO, Badea RA, Stanciu M, Brumaru A. Blockchain-Based Image Copyright Protection System using JPEG Resistant Digital Signature. *2020 IEEE 26th International Symposium for Design and Technology in Electronic Packaging (SIITME)*. 2020;26:206–210. Available from: https://doi.org/10.1109/SIITME50350.2020.9292296.
15) Geethanjali D, Priya R, Bhavani R. Protected Copyright Information Sharing using Blockchain for Digital Garment Design Work. *Journal of Education: Rabindrabharati University*;XXIII, No.:2021–2021.