IPFS-Blockchain-based Authenticity of Online Publications

Conference Paper · June 2018

CITATIONS
0

READS
308

2 authors:

Khaled Salah
Khalifa University
201 PUBLICATIONS **1,242** CITATIONS
SEE PROFILE

Haya Hasan
Khalifa University
8 PUBLICATIONS **4** CITATIONS
SEE PROFILE

IPFS-Blockchain-based Authenticity of Online Publications

Nishara Nizamuddin, Haya R. Hasan, Khaled Salah

Department of Electrical and Computer Engineering
Khalifa University of Science, Technology and Research
Abu Dhabi, UAE

{nishara.nizamuddin,haya.hasan,khaled.salah}@kustar.ac.ae

Abstract. In this paper, we propose a solution to provide originality and authenticity of published and posted freely online digital content such as books, music, and movies. Our solution utilizes a blend of newly emerging technologies that primary include (InterPlanetary File System) IPFS and blockchain smart contracts. IPFS is used to store digital content with a high integrity and global accessibility to all, and Ethereum smart contract is used to govern, manage, and provide traceability and visibility into the history of digital content from its origin to the latest version, in a manner that is decentralized and globally accessed with high integrity, resiliency, and transparency. In the paper, our solution is focused on online book publication, but the solution can be a framework that can be easily extendible and adoptable for, to other digital and multimedia content. The full code of our smart contract is provided, with discussion on implementation and testing of its key functionalities.

Keywords: Ownership Authenticity, Originality, Online Publishing, Blockchain, Ethereum, Smart Contracts.

1 Introduction

The Internet and the digital era have unleashed the unique access to information. With the increased ease in information access and sharing, the authenticity of freely posted and published digital materials is always questionable. The authenticity of digital content is a major challenge for today's online book publishing industry, and digital content in general, as those of multimedia, movies, music, etc. Digital content, available on the internet, during its lifetime can be modified, copied, reproduced, translated into different languages, re-published, and reformatted. There is an immense need for an appropriate authenticity with the ability to trace and track the publication history of posted online material to the original author, writer, or artist, with high degree of trust, credibility, and integrity.

In practice, a hardcopy manuscript of a book or newspaper article can be printed, scanned, digitized, and translated into different languages—resulting in multiple versions of the original manuscripts which were published by different publishing entities or individuals. That is, digital content available across various resources such as online journals, e-books, and websites can indeed be subjected to illegitimate alteration that ultimately leads to tainted information access. Also, there is a lack of a strict audit to ensure that the digital book is verifiable, complete and accurate. While e-book is collated and printed from diverse sources, the authenticity and integrity of digital asset is at stake.

Problem Statement. *To date, there exists a lack of authenticity and integrity of digital content made available online. Freely posted and published online digital contents are not tampered-proof and their publication history cannot be easily tracked in a credible, trusted, open, and decentralized manner.* As shown in Figure 1, a book can be published, and re-published by different publishing entities, and thus producing multiple versions of the original book. The book is originally written by the author which can be made available to public users via various resources such as handwritten manuscript, physical print media, e-books and Internet sources. The credibility of a digital document cannot be checked at any point as the publishers are not accountable for the content published. Furthermore, the readers are totally unaware about the accuracy and authenticity of the available e-book. Typical readers usually accept the online versions of digital manuscripts despite being tampered with.

Figure 1 shows the traditional way of public readers accessing digital assets of e-books from available resources. The original work of the author undergoes various stages of publication process before it reaches the end user i.e., the readers. Typically, author writes the book and chooses a main publisher to submit his original work. The main publisher is granted the publishing permission from the author. Figure 1 illustrates the scenario where secondary publishers (P_1, $P_2 \ldots P_n$) request for publishing permission in different versions from the main publisher. The main publisher P grants permission to the requesting publishers upon agreeing to the terms and conditions are accepted by both the parties. The same book can have many editions and versions with different versions being translated in many languages. In today's book publishing industry, a certain online book version cannot be traced back to an original author, as information is usually defragmented and not available to all readers to verify and examine the authenticity and originality of published content.

Fig. 1. Production of different versions of an online book through various publishers

Blockchain is a newly emerging and disruptive technology that can be key in providing a solution to authenticity of digital materials. Blockchain is the underlying technology of the cryptocurrency bitcoin, but now is seen as a distributed ledger that can be accessed globally by anyone to verify stored data and content, with high integrity, resiliency, credibility, and above all traceability [1]. All of this is done in de-centralized manner and without intermediaries. Later, Ethereum smart contracts provided the ability to upload and execute code that carries out business logic to the blockchain [2,3]. The smart contract code resides on a blockchain as multiple functions with unique addresses that can be called by any user of the blockchain.

Blockchain, however is an expensive medium for data storage, especially for large data and digital content. For efficient storage of large data and content, we propose using IPFS file system. IPFS stands for Inter-Planetary File System [4], which is a distributed, decentralized file system and a platform to store data and files with high integrity and resiliency. Fundamentally, IPFS is a peer-to-peer, open source, content addressable globally distributed file system that can be used for storing and sharing large volume of files with high throughput. Our proposed solution makes use of both blockchain smart contracts and IPFS, whereby the digital contents are stored on the IPFS and the IPFS hashes are stored into the blockchain smart contracts to provide traceability and authenticity. Specifically, the hash generated on storing the documents to IPFS, can be stored in the smart contracts effectively and documents can be accessed using the hash. If there is any change in the content of the digital document, the hash changes, to show that the original content was modified and altered.

In this paper, we propose a combined IPFS-blockchain-based solution to solve the authenticity and originality of digital content posted freely on the Internet. In the paper we show how this problem can be solved for online published books, but our solution can be extended and adopted for, to other digital and multimedia content. We show how

our solution has the ability to trace and track the digital content, with its different published versions, back to the certified true copy created by the original author. The main contributions of this paper can be summarized as follows:

- We propose an IPFS-blockchain-based solution and framework for providing authenticity of published online books and digital content. Our solution provides decentralized storage and governance with different versions of the original book being stored, tracked, and traced with high integrity, and resiliency.
- We present and discuss the system design and architecture, along with sequence diagrams to illustrate the interactions among participants which include author, main publishers, secondary publishers, and readers.
- We provide the full code of the smart contract, and discuss key implementation and testing details to demonstrate the proper operations and functionality of the overall system.

The remainder of this paper is organized as follows. Section 2 summarizes the related work. Section 3 presents our proposed solution for improving authenticity of online books. Section 4 describes key aspects of the implementation and testing of the smart contract. Section 5 concludes the paper.

2 Related Work

In this section, we provide a brief background on the existing approaches found in the literature related to authenticity and originality of books in the publishing industry using blockchain technology. Ericsson [5] proposes a blockchain-based system for tracking the origin of digital assets. It is by converting the digital content and books into a binary file and to store the hash on blockchain. This hash is stored generally with an identifier for the owner. The paper focusses on the idea that ownership can be verified by checking the integrity of digital assets at any point of time. This is achieved through the verification by a centralized unit for security of digital documents such as Security Operations Center (SOC) to estimate the legitimacy of digital assets. But the system itself deviates from the decentralized concept as it operates in the presence of a centralized security unit and has a precarious state for breach of integrity. Moreover the file storage is done on a centralized server which can be a single point of failure, corruption, hacking or compromise.

Gaetani *et* al [6] propose a verification ID which can be used for blockchain-based authenticity of digital assets. This ID is inherently a digital block on the blockchain that can be used for verification of the e-document. That is, whenever an ID is added to the blockchain, an identification certifying service combines the public key with the owner's ID and transfers the ownership of the private key to the user. A blockchain centered handshake mechanism is employed to ensure authenticity of the e-document. But this method suffers detrimental effect as it is purely based on the trust factor on the requesting entity. As the identity of the requesting body is not always reliable in the

digital world, the system is not stable enough to provide a secure storage and access of documents online.

The author in [7] proposes a blockchain-based model for publishing online books and for providing integrity of the digital document. The author achieves authorship by storing the book/file hash and the owner's name in pairs. The author argues that by storing the hash of the file and the block timestamp as pairs, integrity of the document/file can be proved. If the content of the file was modified, then its hash will change, and the smart contract won't be able to access the file, therefore proving that the file content was modified. Sun *et* al. [8] describes a framework for evaluating the trust issues when storing online documents in decentralized networks. In this paper, authors present a framework to quantitatively measure trust, model trust propagation, and defend trust evaluation systems against malicious attacks. This system was used to secure adhoc routing and support to unmask malicious node in a decentralized environment but is not yet implemented as a real-world application.

The authors in [9] propose a blockchain-based personal data management system to ensure that document owners have complete authority over their asset. This model features a blockchain-based automated online document access control system thereby eliminating trust in a third party. Blockchain and off–blockchain storage is combined to construct a management platform which precedes to trust based computing. But the work does not describe about the feasibility of storing larger files. Morgan [10] presents the idea of using blockchain technology to prove the existence of a document using the timestamping concept. The author discusses a method where the document is presented to a site which in turn converts the document into a cryptographic hash. The hash generated represents the content of the document. If the original document is presented, same hash will be generated, notifying that the document is authentic. However, if there is any modification of content, the newly generated hash will not match with the previous hash. The legitimacy of the document can be verified, but this system is not focused about the authority of the owner on his/her document.

Acronis Notary system described in [11] is a blockchain-based notary service which aims at providing a solution for timestamping digital documents. As blockchain is a very expensive storage medium for storing large documents, the proposed approach is to send file hashes to the Notary service. This service calculates hash value, based on the received file hashes and saves the new hash obtained, on the Ethereum network. A verification certificate is provided specifying the technical details of the document. Whenever the document is reflected in user interface, it is shown as 'notarized' or 'certified' by Acronis. By doing so, the system gives an assurance to the user that the online document is identical to the original version, on a bit-by-bit basis. This method supports providing notarization for document authenticity and a certification that an e-book existed at some point of time in the chain. But it doesn't clearly state about the author's rights to claim the ownership of digital book in the decentralized environment.

3 Proposed Solution

Our solution is based on using IPFS and smart contracts of Ethereum blockchain. IPFS is used to store the digital content (of the e-book or multimedia files) in a decentralized, distributed manner that is publicly and globally accessible by all through the use of IPFS hashes. This IPFS hash is used by the smart contract of Ethereum blockchain to ensure integrity, originality, and authenticity. The hash value remains the same if the content of the document or e-book remains intact. If there is an alteration of content during the publication stages, the IPFS hash for the book changes, and would then not match the hash stored within the smart contract. Therefore, each participating entity can track back, and verify the accuracy and history of e-books being stored in the file system and be assured that the book accessed is a legitimate copy of the author's work.

3.1 System Architecture and Design

Figure 2 illustrates the overall system architecture and design for automating the online books authenticity, originality, and integrity using IPFS and Ethereum smart contracts. The proposed solution uses smart contracts to trigger events that are logged to notify the participating parties to keep track of events and transaction details. The figure highlights the interactions of the smart contract with main participants that include author, main publisher P, secondary and other publishers, and readers. The participants of the smart contract can be summarized as follows:

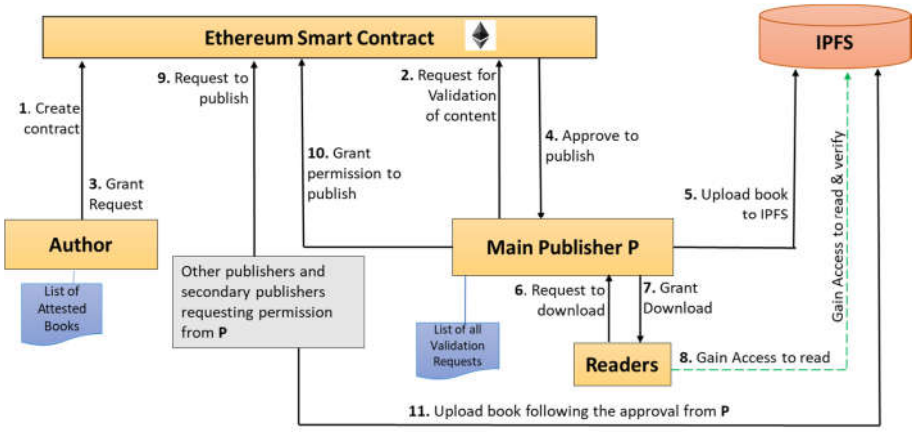

Fig. 2. An overview of the system architecture for automating the online books authenticity using IPFS and Ethereum smart contracts.

- **Author**: The author or artist is the person who owns the IP and original work of the book or digital content. The author creates the smart contract and provides permission for one or more publishers to publish his digital content. The main publisher is also involved in validating and notarizing the book content presented to it by the

publishers. The publisher gets to upload the content on IPFS, only if the notarization by author is successful. The author also maintains the original hash of the book.

- **Publisher**: The publisher is the entity which obtains permission from the author to publish the manuscript content in various sources such as web pages, printed books and e-books. The main responsibility of a publisher is to maintain the digital content intact as presented by author. /Also, a publisher uploads the digital content to IPFS and the IPFS hash is stored in the smart contract with the author's validation and verification.
- **Other and Secondary Publishers**: A book can be translated into different languages and can have different versions or editions produced by different publishers who produced the book in agreement with the original author or with the main publisher.
- **Readers**: Readers are those who request access for legitimate digital books. The smart contract provides the readers with trace back functionality to verify the originality of the book. The readers can access the history of notarizations regarding the originality, authenticity, and integrity of the book.

In our proposed solution, the author initially writes the book, creates the contract which includes key attributes about the book to include book title, original hash of the book and author details. Offline, a main publisher seeks publishing permission from the author after finalizing publishing terms. The publisher, then requests for an approval from the original author before uploading the book to IPFS, which returns a hash. The author then examines the stored IPFS file (with the given IPFS hash) to the original content submitted to the main publisher. The author concludes that the digital manuscript is uncorrupted and attests the publisher copy. The author has a list of books he attested, and also the publishers have a list of book versions attested by them.

Readers or other secondary publishers can also request for content validation history, to know about authenticity of digital manuscript. Once the book is made available on the IPFS, a reader or a secondary publisher can request for history of validation proof. This can be needed to make sure that the document has the notarization of the main publisher from whom publishing permission was granted and the attestation of the original author who validated for the main publisher to publish the content. The smart contract has the ability to display the list of attestations for that the e-book along with the address of the requesting entity, address of the publisher who provided access for the document and the address of the original author. The validation history helps to track and trace back the history of attestation provided for the digital documents.

4 Implementation and Testing

Our smart contract was implemented and tested using Remix IDE http://remix.ethereum.org. In this section, we provide the implementation details and focus primarily on testing the correct interaction and functionality among system participants.

Remix IDE offers rich features that make it possible to test and debug smart contracts prior to deploying them.

4.1 Implementation Details

The code was written in Solidity using the web browser-based IDE, Remix. There are three entities participating in the contract, author, publisher, readers / secondary publishers. Each of the entities has an Ethereum address and can participate by calling functions within the smart contract at certain time stamps. Figure 3, illustrates the message sequence diagram for granting publishing permission with a successful attestation and a failed attestation by author. It shows the interaction between publisher, author and smart contract.

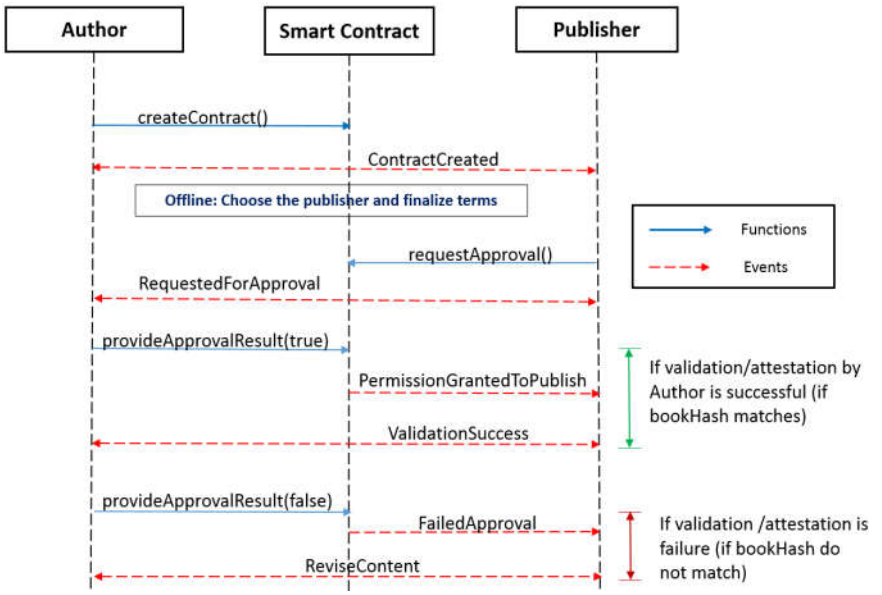

Fig. 3. Message sequence diagram showing scenarios of successful validation and a failed validation.

Figure 4 represents the message sequence diagram for successful and failed trace back of validation history between the requesting readers or secondary publishers, main publisher and the author. Figure 4 illustrates the flow of `traceBackHistory()` function and events `ValidationHistorySuccess` and `FailedValidationHistory` on successful and failed trace back history of attestations respectively.

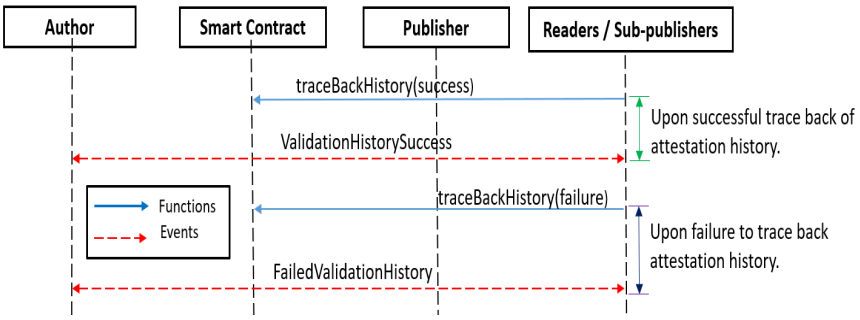

Fig. 4. Message sequence diagram showing scenarios of successful validation trace back and a history of failed validation trace back.

Next, we show the important code snippets of our smart contract. The smart contract code is available at Github To track the state of publishers and their approval results we use mapping, which represents a key-value pair. We also maintain a mapping to record a list of approvals by author and the hash provided by publishers during attestation process. Figure 5 shows mapping where every Ethereum address points to the address of publishers who submitted request for approval and hashes provided by publishers. Figure 5 also shows a mapping which consists of a list of books approved by authors and a mapping consisting of state of publishers.

Fig.5. Mapping of publishers and books attested by author[1].

```
mapping (address=> bool) public recordList;//addresses of publishers and results (true or false)
mapping (address => bool) public approvedAuthors ;
mapping(address=>string) public bookHashes; //hashes provided by publishers
mapping(address=>publisherState) public publishers;
```

Figure 6 represents constructor `OnlineBooksAuthenticity()` which shows that author is the owner of contract. It comprises of initialization such as the `authorName`, `bookInformation` and most importantly consists of the original book hash. The state of contract initially is `NotReady`. The author creates the contract and executes the `CreateContract()`[1] function and invokes `ContractCreated` event.

```
//constructor
function OnlineBooksAuthenticity(){
    bookInformation = "Work of Fiction";
    author= msg.sender;
    authorName= "Danielle Steel";
    IPFShashAuthor= "QmXgm5QVTy8pRtKrTPmoWPGXNesehCpP4jjFMTpvGamc1p";

    contState = contractState.NotReady;
    numberOfRequestsByPublishers = 0;
    numberOfApprovalsByAuthor = 0;
```

Fig.6. Constructor of the smart contract `OnlineBooksAuthenticity()` code[1]

Figure 7 demonstrates the `requestApproval()` function for publisher requesting attestation from the author. At this stage, the state of the contract is `Created` and the publisher state will be `ReadyToSubmit`. The publisher state changes to `SubmittedForApproval` and the contract state changes to `WaitingToProvideApproval`. The event `RequestedForApproval` is triggered and the publisher waits for the approval results (True/False) from the author.

```
function requestApproval(address publisherAddress, string bookHash) NotAuthor  {
 require(contState==contractState.Created && publishers[publisherAddress] == publisherState.ReadyToSubmit);
      publishers[publisherAddress] = publisherState.SubmittedForApproval;
      contState = contractState.WaitingToProvideApproval;
      bookHashes[publisherAddress] = bookHash; //update the mapping
      RequestedForApproval(msg.sender, "Attest and Validate document to proceed for publishing");
      numberOfRequestsByPublishers += 1;
  }
```

Fig.7. Smart contract function for publisher requesting attestation from author[1]

In this paper, we have considered two scenarios based on the approval results provided by the author for every publisher requesting an attestation or validation before uploading the book to IPFS. We have considered two other scenarios based on the attestation history results requested by either readers / secondary publisher. The four scenarios are as follows:

1. If the book validation results i.e. `provideApprovalResult ()` is "True", the transaction is successful, and the smart contract directs the publisher to upload the authentic book to the IPFS and the book is now freely available for readers to access.
2. If `provideApprovalResult()` yields "False", the author does not approve the online book content as the hash submitted by publisher while requesting for attestation does not match the original hash of the book which is stored in smart contract. The change in hash clearly indicates, that the original content was changed / modified. The author triggers an event soliciting the publisher to amend the content as per original document and to resubmit for attestation.
3. If the attestation history trace back i.e. `traceBackHistory()` is successful, the list of attestation made for the book is provided to requesting entity.
4. When `traceBackHistory()` is a failure, it indicates that the book accessed is not validated and hence the requesting entity is provided with the detail that the book is not an authentic work of author.

4.2 Testing and Validation

In this section, we test the correctness of the interaction among participants, and the correct functionality of the overall system. Testing of the smart contract ensured that the flow of the contract followed the correct sequence, validations and refusals of e-book submitted were tested correctly and the attestation history tack back functionality

executed correctly. In this paper, the smart contract is tested for 4 different scenarios. Figure 8 shows the various testing scenarios carried out. The first test group is on the approval from author to publish the book. It is further classified for testing two scenarios – Validation success and Failed Validation. The second test group represents readers or secondary publishers requesting for attestation history details. Test case 2 has two scenarios: Successful traceability, and Failed traceability. All the test cases are discussed in detail in the next subsection.

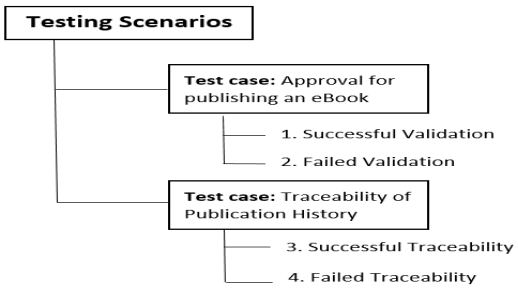

Fig. 8. Smart contract testing scenarios

4.2.1 Test Case 1

Firstly, we test the smart contract for a successful validation scenario. For testing purpose, we consider the Ethereum address of the author to be `0xca35b7d915458ef540ade6068dfe2f44e8fa733c`, the addresses of publishers who submit the request for attestation or validation to be `0x14723a09acff6d2a60dcdf7aa4aff308fddc160c` for Publisher1 and `0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db` for Publisher2. We tested, as shown in Figure 9, the case when the publisher request for a validation to publish from the author. The event `RequestedForApproval` is triggered and Publisher1 waits for the author to attest its content.


```
[
	{
		"topic": "0b7bbff0dc5fdbec48d0ad182e48548f4c5606c90668bec82ac980eceaf2f773",
		"event": "RequestedForApproval",
		"args": [
			"0x14723a09acff6d2a60dcdf7aa4aff308fddc160c",
			"Attest and Validate document to proceed for publishing"
		]
	}
```


Fig. 9. Logs showing event `RequestedForApproval` triggered[1]

The original hash of the book is "`QmXgm5QVTy8pRtKrTPmoWPGX-NesehCpP4jjFMTpvGamc1p`". Firstly, we test for the scenario of successful attestation/ validation of book by the author with Publisher1. Upon successful attestation, events `PermissionGrantedToPublish` and `ValidationSuccess` are triggered and Publisher1 is granted the permission to upload the book on IPFS. Figure 10 shows the scenario of successful validation provided to Publisher1 by author.

```
{
        "topic": "9ecdc72629544c57e07d4ad9ad06b1752ad2aa1fd012b492e2648bc52547648e",
        "event": "PermissionGrantedToPublish",
        "args": [
                "0xca35b7d915458ef540ade6068dfe2f44e8fa733c",
                "Content Verified by Author. "
        ]
},
{
        "topic": "6a8631a55e9a051559f93898d7bab45981eb82c36fecbc77256bc93af8826dfe",
        "event": "ValidationSuccess",
        "args": [
                "0x14723a09acff6d2a60dcdf7aa4aff308fddc160c",
                "Proceed to Publish Content on IPFS"
```


Fig.10. Logs showing a successful attestation[1]

4.2.2 Test Case 2

Secondly, we test the smart contract for the scenario of a failed validation with Publisher2. The validation fails because of change in hash provided by Publisher2 while requesting attestation to publish. The author refuses approval and asks Publisher2 to resubmit the document after making the necessary amendment. We tested a case, where Publisher2 submitted a different hash value - "`QmYh1A5qcUXXAxE-PaVmY7Frgw6rGwyUqE5uc71ThtuoAUM`". Figure 11 shows that the approval failed, and the author recommends for amendment of content before resubmitting for attestation again. Events `FailedApproval` and `ReviseContent` are invoked to notify Publisher2 to amend the book content.


```
        {
                "topic": "159da0d7bb81fb987a7b3facd641bdbec279ff4b036a257fc585d8ef94d5052c",
                "event": "FailedApproval",
                "args": [
                        "0xca35b7d915458ef540ade6068dfe2f44e8fa733c",
                        " Content Modified / Corrupted: Hash does not match . Failed to be approved by Au
thor"
                ]
        },
        {
                "topic": "78ac3d60d60fbf6cbd0cf9816e237d6eba02853d97d20da8fc8f50094c4dc37f",
                "event": "ReviseContent",
                "args": [
                        "0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db",
                        " Amend content and request for attestation again."
```


Fig.11. Logs showing a validation failure with events `FailedApproval` & `ReviseContent` triggered[1]

4.2.3 Test Case 3

Next, we tested for successful traceability of the attestation history of the document with a reader whose Ethereum address is `0x583031d1113ad414f02576bd6afabfb302140225`. The reader requests to

prove that the document is the original work of the author and has a notarization proof. Figure 12 represents the scenario where the validation history of the document is obtained successfully displaying the address of Publisher1 from whom the reader obtained book access, the address of the original author. Event `ValidationHistorySuccess` is triggered and the smart contract assures the reader that the received copy is a legitimate work. The logs also show that the book was initially attested by author and the readers can confirm that the received book is authentic.

Fig. 12. Logs showing event `ValidationHistorySuccess` triggered to show the content validation history of the book[1]

4.2.4 Test Case 4

Lastly, we test for the scenario of attestation trace back failure with `0xdd870fa1b7c4700f2bd7f44238821c26f7392148`, the address of the secondary publisher who requests for attestation history trace back from another publisher `0x583031d1113ad414f02576bd6afabfb302140225`. Figure 13 shows the testing results. Event `FailedValidationHistory` is invoked as the attestation is not done by the actual author. This event indicates that the online book requested by the secondary publisher is not attested by the Publisher2 to whom the request for validation history was made.

Fig.13. Logs showing event `FailedValidationHistory` invoked[1]

5 Conclusion

In this paper, we have proposed a solution to solve the originality and authorship authenticity of freely published and posted online books and documents using IPFS and Ethereum smart contracts. Our solution is focused on authenticity of online books, but the solution in terms of architecture, design, sequence diagram, logics, smart contract code, and overall aspects are generic enough to be easily extended and used to provide the originality and authenticity, as well as integrity, to all other forms of digital assets. We implemented and verified the functionalities of the smart contract code using Remix IDE. As a future work, we plan to deploy the smart contracts on the real IPFS and Ethereum network and develop frontend Decentralized Applications (DApps) with different views to authors, main publishers, secondary publishers, and readers.

References

1. S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system”. (White paper). 1
2. K. Toyoda et al., “A Novel Blockchain-Based Product Ownership Management System (POMS) for Anti-Counterfeits in The Post Supply Chain, “in IEEE Access, vol. PP, no.99, pp. 1-1.
3. ‘Ethereum’, http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html/, last accessed 20 December 2017.
4. M. Pors, “Understanding the IPFS White Paper part 2”, https://decentralized.blog/understanding-the-ipfs-white-paper-part-2.html, last accessed 2018/3/18.
5. Ericsson Home Page, Ericsson White Papers – “Industrial Blockchain and Data Integrity”, https://www.ericsson.com/hyperscale/cloud-infrastructure/data-centric-security/data-integrity-assurance, last accessed 2018/2/16.
6. E. Gaetani , L Aniello, R Baldoni , F Lombardi , A Margheri, and V Sassone,” Blockchain-based Database to Ensure Data Integrity in Cloud Computing Environments”, 1st Italian Conference on Cyber Security, ITASEC17, pp 146-155, Venice, Italy (2017).
7. N. Prusty, “Building Blockchain projects – Develop real-time practical DApps using Ethereum and JavaScript”, 1st edition, Packt Publishing Ltd, Birmingham, UK (2017).

8. Y. Sun, Z. Han, W Yu, K.J.R Liu,"A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks," INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pp1-13, Spain (2006).
9. G. Zyskind, O. Nathan, A Pentland, "Decentralizing Privacy: Using Blockchain to protect personal data", Security and Privacy Workshops (SPW), 2015 IEEE Conference Proceedings, pp 180-184, San Jose, CA, USA (2015).
10. P. Morgan, "Using Blockchain Technology to Prove Existence of a Document", https://brave-newcoin.com/news/using-blockchain-technology-to-prove-existence-of-a-document/ last accessed 2018/2/20.
11. Acronis Inc., "Acronis Notary: a new way to prove data authenticity via blockchain", https://www.acronis.com/en-us/articles/data-protection, last accessed 2018/2/21.