This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905689, IEEE Access

Combating Deepfake Videos Using Blockchain and Smart Contracts

Haya R. Hasan and Khaled Salah

***Abstract*—With the rise of AI and deep learning techniques, fake digital contents have proliferated in recent years. Fake footage, images, audios, and videos (known as deepfakes) can be a scary and dangerous phenomenon, and can have the potential of altering the truth and eroding trust by giving false reality. A Proof of Authenticity (PoA) of digital media is critical to help eradicate the epidemic of forged content. Current solutions lack the ability to provide history tracking and provenance of digital media. In this paper, we provide a solution and a general framework using Ethereum smart contracts to trace and track the provenance and history of digital content to its original source–even if the digital content is copied multiple times. The smart contract utilizes hashes of the InterPlanetary File System (IPFS) used to store digital content and its metadata. Our solution focuses on video content, but the solution framework provided in this paper is generic enough and can be applied to any other form of digital content. Our solution relies on the principle that if the content can be credibly traced to a trusted or reputable source, the content can then be real and authentic. The full code of the smart contract has been made publicly available at Github.**

***Index Terms*—AI, Deepfake, Blockchain, Ethereum, Smart Contracts**

I. Introduction

THE recent rise of AI, deep learning, and image processing have led the way to the production of deepfake videos [1], [2]. A video as short as one minute of the former U.S. President Barack Obama went viral in April 2018, in which Obama was seen to say things he never said [3]. Deepfake videos are dangerous, and can have the potential to undermine truth, confuse viewers and accurately fake reality. With the advent of social networks, proliferation of such content can be unstoppable and can potentially exacerbate problems related to misinformation and conspiracy theories. In some early examples of deepfakes, a large number of famous political leaders, actresses, comedians, and entertainers had their faces stolen and weaved into porn videos.

Deepfake videos are far more realistic and much easier to make than traditional Hollywood-like fake videos which are typically done manually using image manipulation tools like Adobe Photoshop. Deepfake videos make use of deep learning techniques with input of large samples of video images to achieve face swapping. The higher the number of samples, the more realistic the outcome becomes. The Obama video was fed with more than 56 hours of sample recordings in order to make it extremely real and believable [4]. Deepfake videos were not a huge concern at the beginning when they first appeared targeting celebrities. The authors in [5], [6] describe deepfake videos as a data catastrophe and called for incentivizing the general public to make good use of new technologies and to post ethically and responsibly digital content on social media outlets.

It is crucial to have techniques to detect, fight, and combat deepfake digital content that may include fake videos, images, paintings, audios, and so on. Achieving this purpose is not difficult if there is a credible, secure, and trusted way to trace the history of digital content. Users should be given access to a trusted data provenance of the digital content, and be able to track back an item in history to prove its originality and authenticity [6]. This mechanism can help assist users from being tricked or lured into believing in fake digital content.

Current solutions are available to prove the authenticity of physical (and not digital) artwork. For instance, a certificate of authenticity (COA) is given with the purchase of an artwork. Moreover, it is possible to forge this certificate or to find it unsigned from a known and trusted authority. Moreover, artwork bought from a secondary market is much harder to prove its origin. The only approach currently sought, is by manually asking the gallery or the product source for the COA they have from the previous owners as well as their receipts [7]–[9]. In a way, the buyer is left with substantial manual work and checking to do to achieve accurate artwork provenance.

As of today, there are no established methods for checking the originality of an online posted or published digital video, audio, or image. The idea to subject such digital content to a COA is not feasible. It is extremely difficult to determine in a credible and trusted way the true origin of a posted digital item. A typical user usually uses online search engines to try to find relevant posts, blogs or reviews on the digital media to judge its authenticity. Hence, there is an immense need for a Proof of Authenticity (PoA) system for online digital content to identify trusted published sources and therefore be able to combat deepfake videos, audios, and images.

In this paper we present a decentralized Proof of Authenticity (PoA) system using the disruptive technology blockchain. Blockchain has the ability to provide immutable and tamper-proof data and transactions in a decentralized distributed ledger [10]. Blockchain applicability is immense, and the technology poised to transform and impact across many businesses, industries, and domains as those in finance [11], the food industry [12], supply chain management [10], [13], health management [14], IoT [15], to name just a few.

Blockchain has capabilities to provide key features that can

H. Hasan is a research assistant at Khalifa University, Department of Electrical and Computer Engineering, UAE, Abu Dhabi, email: haya.hasan@ku.ac.ae

K. Salah is a professor at the Electrical and Computer Engineering Department, Khalifa University, UAE, Abu Dhabi, email: khaled.salah@ku.ac.ae

be utilized for proving authenticity and originality of digital assets in a way that is decentralized, highly trusted and secure. [16], [17], with tamper-proof records, logs, and transactions which are openly accessible to all in case of permissionless blockchain, or restricted to certain participants in case of permissioned blockchain. For deepfakes, the permissionless or public blockchain is the most suitable. We base our solution in this paper on the public Ethereum blockchain with smart contracts to govern and capture the history of transactions made to digital content [18].

In this paper, we propose a blockchain-based solution and a generic framework for the proof of authenticity of digital assets that may include videos, audios, images, etc. Our solution allows for publicly accessible, trusted, and credible data provenance, with tracking and tracing history of a published online video. Our solution focuses on video content, but the solution framework provided in this paper is generic enough and can be applied to any other form of digital content as audios and images. The primary contributions of our paper can be summarized as follows:

- We present an Ethereum blockchain-based solution that establishes authenticity of digital content by providing credible and secure traceability to a trusted artist or publishing source. Throughout the paper, the term ”artist” is referred to the creator or publisher of the digital content. Artists can include freelance or employed photographers, paparazzi, journalists, reporters, etc.
- We present the system architecture and design details with entity relations, sequence diagrams, and algorithms used for Ethereum smart contracts to control and govern interactions and transactions among participants.
- We integrate into our blockchain-based system design key features of the InterPlanetary File System (IPFS) decentralized storage [19] and reputation system, Ethereum Name service, as well as other off-chain resources to access an artist's profile.
- We present the full implementation smart contract code[1] as well as testing details.
- We provide testing details to show the correct system functionality. We also provide a discussion on cost estimation and security analysis of our solution.

The remainder of this paper is organized as follows. Section II provides the related work. Section III presents the proposed blockchain solution. Section IV describes the implementation and testing details. Section V discusses the cost and security analysis of the implemented solution and Section VI concludes the paper.

II. Related Work

In this section, we review and discuss related work found in the literature on authenticity and the originality of digital assets and deepfake content.

The authors in [20] proposed a method to detect deepfake videos using Artificial Intelligence (AI). The proposed method depends on an AI algorithm fighting another AI algorithm. Their technique relies on training convolutional neural networks (CNN) with manipulated and real figures. Testing was carried out using four different CNN networks with varying accuracy results between 84% to 99%. Their results look promising, however, the authors stated many challenges that yet remain to be solved. The presence of glitches in the currently obtained deepfake videos make their method give positive results. Therefore, they reckon that deepfake videos with a high resolution and quality will be hard to detect [21].

A US-based startup company called Truepic [22] has developed a system involving mobile apps for typical users and freelancers for capturing images and saving them to the company's servers. The purpose of saving the images is to preserve their integrity. Hence, any forgery attempt can be easily discovered by comparing it with the image from the servers. They hope that in the future their technology will be used in collaboration with other social media parties that will verify any uploaded images with the images in the Truepic's servers and any change would therefore, be detected. Truepic also uses blockchain to store metadata of saved images to ensure immutability. This method relies heavily on trusting Truepic with the images and that all the uploaded images are untampered and real. It is not clear how the method works when inserting logos, text tickers, subtitles, or closed captions within the images or video frames.

Another UK-based startup company called Serelay [22] employs a technique similar to Truepic to eradicate the spread of deepfake videos and images. Serelay also has a mobile app that users need to use while taking their images and videos. It then computes a special unique fingerprint which is saved in its servers, unlike Truepic which saves the whole image. Serelay claims their method would protect the privacy of its users. They also claim that any pixel that is edited in the original image can be detected using the computed fingerprint. The system is centralized. Their method relies solely in trusting that Serelay act honestly and not tamper with the computed fingerprints and results.

An online blockchain based startup called PROVER is specialized in verifying the authenticity of user-created videos [23]. PROVER works by creating a unique hash while the user is capturing the video. The tool is able to remember the unique hashes of the videos. The main objective of PROVER is to help eliminate any forgery of a video. Users can then check the details of the video such as data and time. It is not clear how their method applies to other types of digital content. Also, PROVER does not indicate a method to trace back an edited or re-sold versions of a captured video.

The authors in [24] focus on using blockchain to only ensure the integrity of a video content. Their approach depends on hashing the video and securing the hash on the immutable blockchain. Any manipulations on the video will result in a mismatch in the hash. OriginalMy is another Brazilian based startup that utilizes public blockchain to register and verify the authenticity of digital documents, contracts and identity of people [25]. The approaches in both [24], [25] do not provide a mechanism to trace different versions of a video.

Our proposed framework is built on blockchain's key feature of transparency, traceability and time-sequenced logs

[1] https://github.com/smartcontract694/PoA/blob/master/code

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905689, IEEE Access

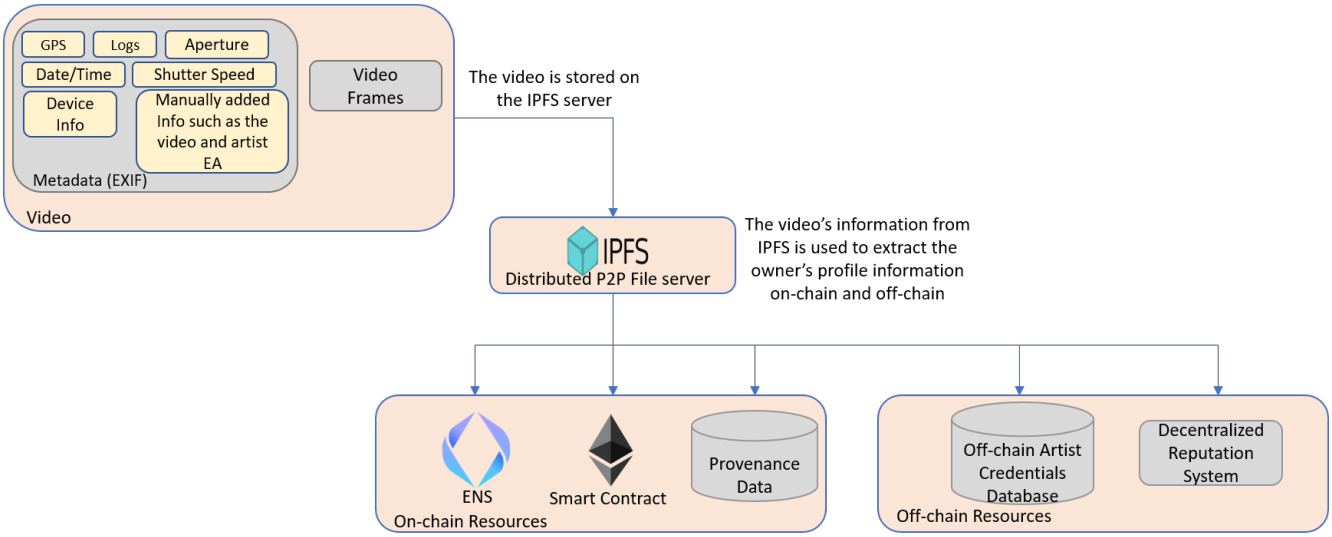

Fig. 1: System overview highlighting key components of proposed solution

to provide a highly secure and trusted history tracking and tracing that may involve multiple versions, in a decentralized manner with no intermediaries or trusted third parties. In this paper, our underlying principle of solving the deepfake problem simply relies on providing undisputed traceability to the original source.

III. Proposed Blockchain-based PoA

This section describes and details our Ethereum blockchain-based approach for proof of authenticity (PoA) of videos. This approach can also be used for other types of digital content such as audios, manuscripts, photos, and images.

A. System Overview and Design

In our solution, all the participating entities, which are the original artist as well as any secondary artist have Ethereum addresses. We further describe key system components as depicted in Figure 1 as follows:

- **Video:** A video has important information other than the video frames as can be seen in Figure 1. Video key attributes are stored as 'Metadata' in an (Exchangeable Image File) EXIF format. The metadata of a video contains information related to the device capturing the video, capture settings, date and time of capture, as well as logs and manually added information that the video creator can add. Every video will be associated with an Ethereum smart contract that can be created by an artist or news source. The Ethereum address of the artist as well as the address of the smart contract are integral parts of the metadata.
- **IPFS Storage:** The video and its associated metadata are stored on a decentralized, content-addressable, peer to peer file system such as the InterPlanetary File System (IPFS) [19]. IPFS generates a unique hash which is the address of a bundle of files containing the video content and its metadata. The hash address is used to locate and access the bundle of files stored on the the IPFS network. Moreover, the IPFS bundle can include a file containing the terms and conditions agreement of copying and editing in case the video is to be copied to create different content by other authors or artists. The IPFS hash generated from the saved form would also be used in the smart contract.
- **On-chain Resources:** After the video's IPFS hash is created, a smart contract is created by the original artist (owner) on the Ethereum blockchain. The contract has attributes and variables to capture the video details and owner's information. It also contains **functions** that enable other secondary artists to request permission to share, edit, and distribute based on the terms and conditions of the agreement form. Moreover, the smart contract contains **modifiers** that restrict access to the methods based on roles or contract state. In addition, **events** are used to create notifications and keep logs about important results and requests. **Variables** are also used to store static information such as the video related data as well as the contract state. Similarly, any edited video by a secondary artist will have its own smart contract with a link to the original video. Hence, all edited videos of any original video are 'child' videos and are available in a list in the original video's smart contract. Therefore, a user who would like to trace a video to its origin can easily do so using the on-chain resources such as the smart contract which has the list of all the children video's smart contracts as well as a link to their parent's smart contract. Therefore, this data along with the logs and notifications created on the ledger are provenance data that can be used by the user for tracking with great transparency.
- **ENS Service:** A user can also make use of the Ethereum Name Service (ENS) [26] as shown in Figure 1. ENS is essentially a distributed name registry system and is used to associate Ethereum address (which is a 20-byte of random values) of artists to a human-readable texts

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905689, IEEE Access

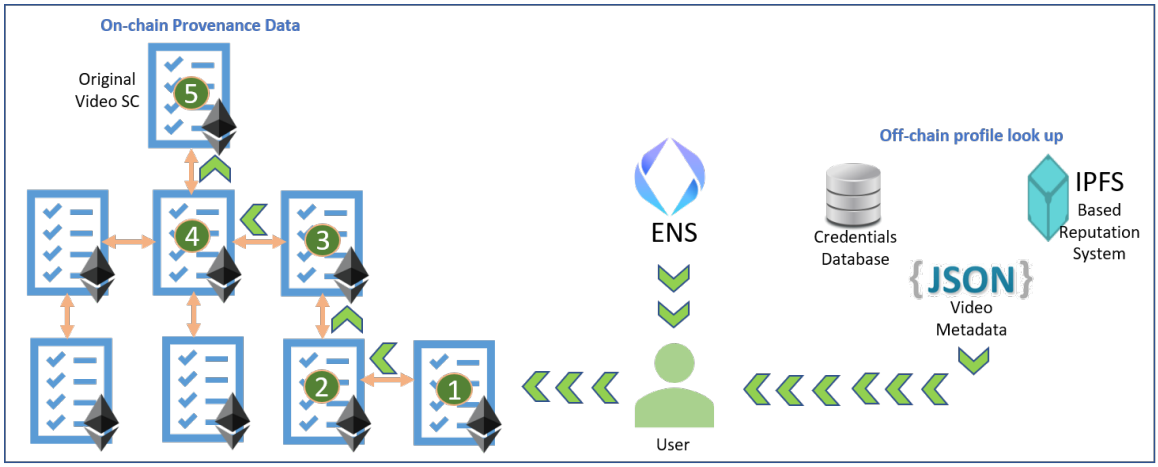

Fig. 2: Tracing video source origin using the proposed solution

capturing the artist's real identity including name, company and profile. This identity is saved in a decentralized way on the distributed ledger.

- **Off-chain Resources:** A user tracing the data can also look at the off-chain resources which are part of our proposed solution. The smart contract Ethereum address of a video and the Ethereum address of the owner can also be linked to an off-chain credentials database as seen in Figure 1. This database contains details about the owner as well as a link to their ENS profile. It will also contain details about other videos the video owner has, in order to provide a full profile and artwork of the artist.
- **Decentralized Reputation System:** Part of the off-chain resources is a decentralized reputation system. The profile of the video creator is linked to the decentralized reputation system that gives a score of the artist's reputation, in addition to review comments [27]. Such system can be built using a combination of smart contracts to calculate the score as well as IPFS to store the review comments. Providing reputation and comments can be open to the public or restricted to voting members. A reputation system becomes important especially for unknown or new artists or freelance photographers or journalists. Voters can give a reputation to an artist as well as to endorse the artist with skills. Hence, the reputation system enables the user to better judge if artists and their contents can be trusted or not.

B. Tracing a Video to its Origin

The main aim of the proposed solution is to assist a user in tracing back a video with multiple versions to its origin. If a video cannot be traced to its original publisher, then it cannot be trusted. Figure 2 shows how a user has interface and accessibility to multiple system components including smart contracts, IPFS, ENS, and other on- and off-chain resources to establish authenticity of the video content. A front-end decentralized application (or user DApp) can be developed for the user to automate the authenticity process, or it can be integrated within video players or web browser to indicate authenticity of played or displayed digital content. In Figure 2, every video is associated with a smart contract that points to its parent video and every parent video is linked to its child, in a hierarchical fashion. As shown in the figure, a can trace smart contract "1" to its parent smart contract "2" which is linked and traceable to smart contract "3". Smart contract "3" points to "4" which is traceable to the original smart contract "5". This provenance data is openly accessible and available to all through the Ethereum ledger.

In addition, the user can also look for the video artist's Ethereum address in the Ethereum Name Service to know more about the artist's profile and information details. Furthermore, the user can also utilize the off-chain resources for a profile lookup. This includes using the information in the video's metadata which is available as a JSON object on the IPFS servers. Moreover, the user can also build a better profile by using the credentials database which would have also a link to other work done by the artist as well as the artist's reputation which is based on an IPFS reputation system as shown in Figure 2.

IV. Implementation and Testing

The smart contract is written in Solidity language, and compiled and tested using the Remix IDE. Remix facilitates for the user to write and execute the codes of a smart contract. It also provides a debugging and testing environment for solidity code. This section discusses the implementation and the testing details.

A. Implementation Details

The owner (original artist) of a video first creates a smart contract where other artists can request a permission to edit, alter or distribute according to the terms and conditions of an agreement form. The agreement form is saved on the IPFS server [19] and its hash is available as an attribute in the smart contract.

The secondary artist requests first permission to edit, alter or share. A request sent by the secondary artist is also a confirmation to the terms and conditions of the agreement

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905689, IEEE Access

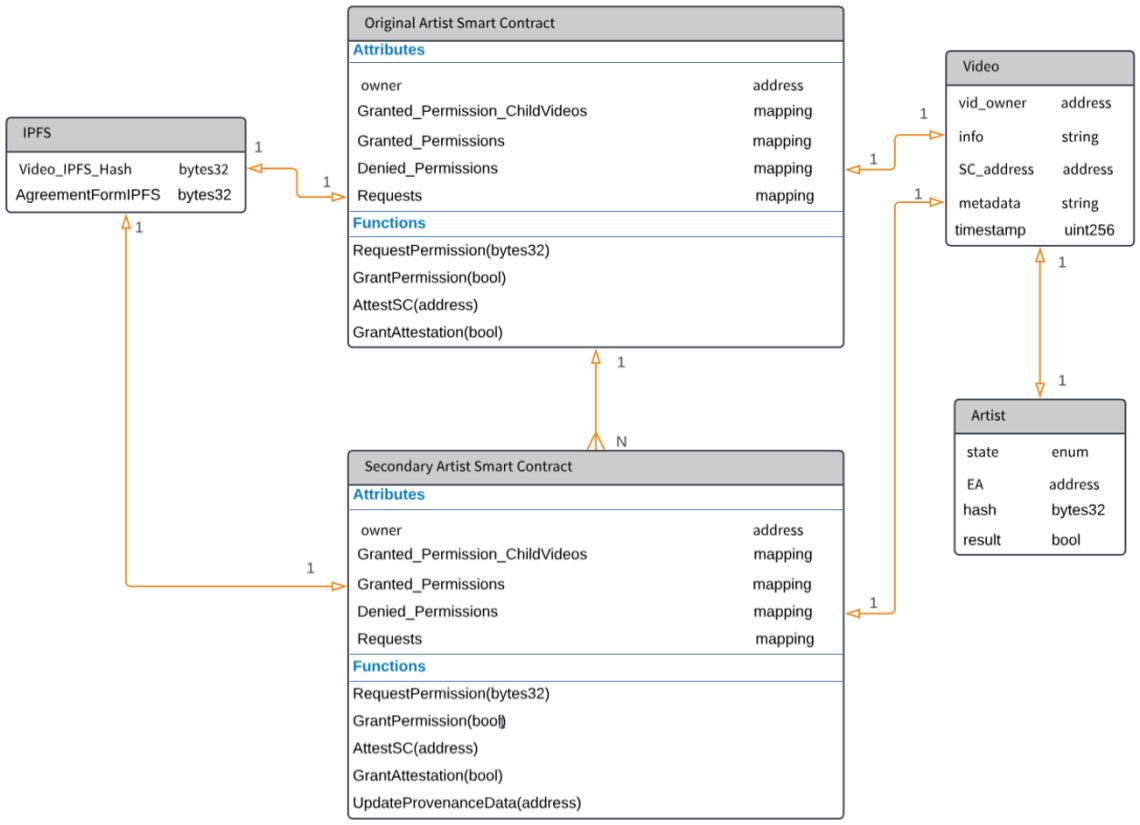

Fig. 3: Entity relationship diagram

form. This request is assessed by the original artist and the result is then announced. The contract can handle multiple requests at the same time and can handle multiple different requests by the same artist. Once an artist gets an approval to their request, they create a child contract which is similar to the original contract and they update the parent's information. The secondary artist then requests the attestation of their newly created contract from the original artist through the contract of the original video. The original artist then approves and grants the attestation after checking the newly created smart contract. A successfully attested smart contract would then be added as a child in the original smart contract. Hence, both the contracts point to each other as each one has the Ethereum address of the other as part of their attributes.

Figure 3 shows the relationship among the different entities of the smart contract. Firstly, the smart contract of the original artist is created using the attributes shown in Figure 3 such as the owner which holds the Ethereum address of the original artist, and the mappings which hold the lists of video details based on their status of granted or denied permissions. Moreover, all requests are saved for reference and history tracking. In addition, an important list which helps in traceability is the list of granted attestation videos which are considered as child videos to the original contract.

Every contract is created for only one video. Hence, the 1:1 relationship between a contract and a video entity. Moreover, every video is linked to only one artist with one Ethereum address. Furthermore, a smart contract can have multiple child contracts based on successful attestations. Therefore, a 1:N relationship between the original artist smart contract and the secondary artists' smart contracts as shown in Figure 3. Lastly, IPFS [19] is an entity also with a 1:1 relation with any smart contract created as every video is uploaded on the IPFS servers and its IPFS hash is an attribute in the smart contract. Moreover, the terms and conditions agreement form of every contract is also uploaded on the IPFS server and its hash is an attribute in the smart contracts created for the videos. Furthermore, The full code[1] is also made available for all the details.

Figure 4 presents the sequence diagram that captures the interactions among the original artist, a secondary artist and the smart contract. The smart contract is owned by the original artist and the secondary artist is interested in requesting a permission to alter, edit and have distribution rights. Hence, as shown in Figure 4 the secondary artist calls the $RequestPermission()$ function which indicates that they have also read the terms and conditions agreement form available on IPFS file server [19]. This creates two successful events announcing the registration of an artist request. The original artist would then reply back with the result of whether to grant the permission or deny it. Based on the result from the original artist three different scenarios take place. Either the permission is denied which is shown in Alternative 1 in Figure 4 or the permission is granted as shown in Alternatives 2 and 3.

A granted permission allows the secondary artist to create

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905689, IEEE Access

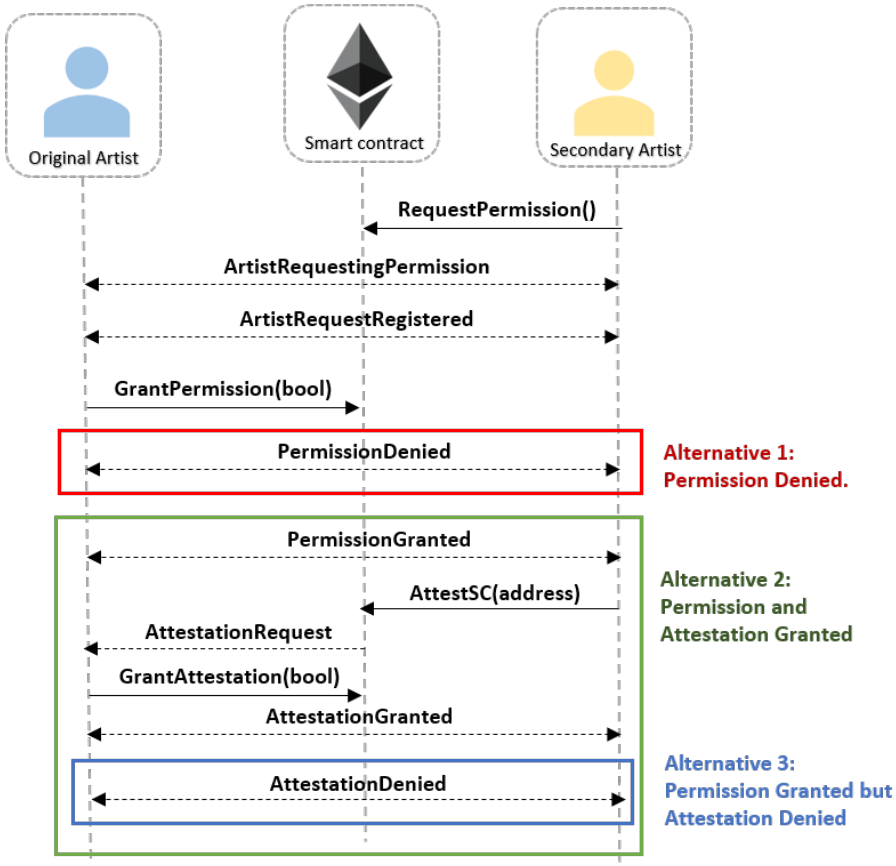

Fig. 4: Sequence diagram showing the function calls and events considering three different scenarios

a child smart contract which is an exact copy of the main contract in terms of function names and attributes. The child contract should have the Ethereum address (EA) of the parent contract. The secondary artist then asks for an attestation using the $AttestSC()$ function available in the original video's smart contract. The original artist will then check the newly created child contract and grant the attestation as shown in Alternative 2 or deny it as shown in Alternative 3 in Figure 4.

1) Requesting Permission: The algorithm for requesting a permission by an artist to edit, modify, and provide distribution rights to other interested artists in the newly modified content is given in Algorithm 1. The smart contract keeps track of all requests sent to it. An IPFS [19] hash of the agreement form is available in the contract. Therefore, a request sent by the artist is a confirmation that the artist has read the terms and conditions and has agreed to them.

The request includes the Ethereum address (EA) of the artist as well as the IPFS hash of the edited video. The code only accepts requests that have not been sent before. This is done by checking the list of requests and their IPFS hashes. If the IPFS hash is not available in the list of requests, a notification event is broad casted to all that a new request has arrived and is waiting. Algorithm 1 explains how the request is added to the list with generation of a notification to announce the completion of the artist registration.

2) Granting Permission: Once a permission request is sent to the contract, the original artist should check the content off the chain and determine if they would like to grant or deny the request. The result can only be sent out by the original artist and is provided after the contract verifies the details of the registered artist. The artist must have sent a request to be granted a permission. Algorithm 2 describes the exact procedure and shows how the result is a boolean.

If the result is true, then the request is updated and added to a list that holds details on videos with granted permission. Otherwise, the video is added to a list of denied permissions.

Algorithm 1: Requesting Permission

Input : R, $IPFShash$, EA

R is the list of all submitted requests to this contract.
EA is the Ethereum Address of the artist requesting permission.
if *new requests* $IPFShash \notin R$ **then**
    Create a notification about artist with EA requesting permission.
    Update the R list by adding a new entry with the artist state, EA, hash and request result.
    Create a notification about the completion of the registration of the artist with EA.
end
else
    Revert contract state and show an error.
end

This information is available for the public in the smart contract, to help ease traceability and make things transparent. At the end a notification is created about the result.

Algorithm 2: Granting Permission

Input : GP, DP, R, secondary artist, result, caller, original artist

GP is a list that holds information on videos with granted permission.
DP is a list that holds information on videos with denied permission.
R is the list of all submitted requests to this contract.
$original\ artist \leftarrow original\ artist\ Ethereum\ address.$
if $caller == originalartist \wedge (state \in secondaryartist) == Sent\ Request$ **then**
  if *result == True* **then**
    Add new video in GP list with all the video details.
    Update the result of the video in the R list.
    Create a notification about the video with granted permission.
  end
  else
    Add new video in DP list with all the video details.
    Update the result of the video in the R list.
    Create a notification about the video with denied permission.
  end
end
else
  Revert contract state and show an error.
end

3) Updating Provenance Data: A secondary artist with a video that was granted permission then creates a new child contract that holds the details of the newly created video. In addition to that, the newly created contract should also point to its parent. Hence, the original video should also be part of the child contract along with its details and smart contract address as agreed per the terms and conditions form. Algorithm 3 shows the details of the procedure and how it is done in the newly created child contract of the edited video.

4) Attesting a secondary artist's Smart Contract: If an artist is granted the permission, they should then create a child smart contract as per the terms and conditions of the agreement form and update the parent smart contract to the original one by adding the parent's EA and other information in the child contract. The secondary artist would then request the original creator to attest their smart contract by providing the details of the contract such as the EA of the contract and the details of the video.

Algorithm 4 shows the full details of the attestation process. The original artist provides the attestation outcome as true or false. If the outcome is true, the newly created smart contract is added as a child contract in the parent's list. This list holds all videos that have granted permissions and successful attestation. Therefore, at the end, a successfully attested video

Algorithm 3: Updating the Provenance Data in a Child SC

Input : hash, metadata, SCaddress, EA, caller, secondary artist

$hash$ is the IPFS hash of the parent video.
$SCaddress$ is the Ethereum address of the parent contract.
$subartist$ is the Ethereum address of the sub artist who owns the child contract.
$metadata$ is the metadata of the parent video.
EA is the Ethereum address of the original artist.
if $caller == secondaryartist$ **then**
  Update the $metadata$ of the parent video.
  Add the $hash$ to the parent video details.
  Add the $SCaddress$ to the parent video details.
  Update the EA of the owner of the parent video.
end
else
  Revert contract state and show an error.
end

Algorithm 4: Attesting a secondary artist's Smart Contract

Input : CV, result, hash,information, metadata, SCaddress, caller, secondary artist

CV list of all child videos with granted permission and attestation .
$SCaddress$ is the Ethereum address of the child contract.
$subartist$ is the Ethereum address of the sub artist who requested previously the attestation.
$result$ is the attestation result from original artist.
if $caller == original\ artist \wedge (state \in secondaryartist) == GrantedPermission$ **then**
  if $result$ *== True* **then**
    Create a new entry in CV.
    Add to the newly created entry the video $information$, $metadata$, $SCaddress$, $sub\text{-}artist$. Create a notification that a new child video is added to the CV list.
    Update state of secondary artist in smart contract as "Approved".
  end
  else
    Update state of secondary artist in smart contract as "Denied".
    Create a notification about the denied attestation.
  end
end
else
  Revert contract state and show an error.
end

will hold the EA of its parent contract and the parent contract would have all EA of its children. Hence, the contracts will be pointing and linked to each other.

B. Testing and Validation

This section presents the details of testing the smart contract using Remix IDE in-browser developing and testing environment. The section describes testing key functions showing the corresponding outputs and logs. For our testing scenarios, we assume two participants interacting with the smart contract. The original artist has the Ethereum address (EA) "0xca35b7d915458ef540ade6068dfe2f44e8fa733c" and the secondary artist has the EA "0x14723a09acff6d2a60dcdf7aa4aff308fddc160c". All functions have a state requirement or a condition that has been tested successfully. The state of the smart contract reverts back to this original state if any of the conditions are not met.

1) Permission Request: The secondary artist sends a request to the smart contract to ask for permission to edit, alter and have distribution rights. This is done using the $RequestPermission()$ function.

Figure 5 shows the logs after the secondary artist has sent the request. The request includes the IPFS [19] hash of a video that will help the original artist to determine whether to grant the secondary artist a permission or not. A successful request leads to the generation of an event to notify the original artist about the sent permission request as shown in Figure 5.


```
"from": "0x692a70d2e424a56d2c6c27aa97d1a86395877b3a",
"topic": "0x85ace08f038481adc2bfd0f855a5103eed84d76d1889e2837c00a5f0b64dd50b",
"event": "ArtistRequestingPermission",
"args": {
        "0": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
        "artist": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
        "length": 1
}


"from": "0x692a70d2e424a56d2c6c27aa97d1a86395877b3a",
"topic": "0xeb56067adf2919f9de03c1d1b99d640dc3897790d2af7784e06c9299b66b27f0",
"event": "ArtistRequestRegistered",
"args": {
        "0": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
        "1": "0x914a2bac6a231a389d48f1542e97f7d201131a91ff3fa839aa2cb6ac488b78f1",
        "artist": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
        "IPFS_Hash": "0x914a2bac6a231a389d48f1542e97f7d201131a91ff3fa839aa2cb6ac488b78f1",
        "length": 2
}
```


Fig. 5: Logs showing a request sent by a secondary artist

2) Permission Granting: It is important to check whether the original artist is able to grant permission to a request. Therefore, granting a permission was successfully tested and the logs can be seen in Figure 6.

The original artist grants a permission by sending a boolean to the smart contract. In this case as shown in Figure 6, the owner (original artist) has granted the permission successfully by sending a $true$ to the smart contract. Hence, a notification is sent out to announce a successful permission grant to all the participating entities.

3) Update Provenance Data: When a secondary artist is granted the permission to edit and distribute, then they should create a child contract, update the parent information in their smart contract and then send an attestation request.


```
"from": "0x692a70d2e424a56d2c6c27aa97d1a86395877b3a",
"topic": "0x30f499dee5aca26bf88e6932509f2693b637e648ec36ac97034db8869f118639",
"event": "PermissionGranted",
"args": {
        "0": "Permission Granted to address ",
        "1": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
        "info": "Permission Granted to address ",
        "artist": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
        "length": 2
}
```


Fig. 6: Execution of a successful permission grant

Updating the parent information is vital to ensure traceability is possible by other users. In our testing scenario, the secondary artist created a smart contract with the address "0x1439818dd11823c45fff01af0cd6c50934e27ac0". Using the functions of the smart contract, the parent information in the newly created contract were also updated. This includes the important information about the main original video as well the smart contract address of the parent which in this case is "0x692a70d2e424a56d2c6c27aa97d1a86395877b3a".

Figure 7 illustrates the scenario when the parent successfully added in the newly created contract of the secondary artist.


```
"string information": "Comedy",
"bytes32 hash": "0x785b2bac6a231a389d48f1542e97f7d201131a512f3fa839aa9cb6ac488b78f6",
"address SC": "0x692a70D2e424a56D2C6C27aA97D1a86395877b3A",
"address EA": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c"
```


Fig. 7: Details of the original video (parent) contract successfully added in the newly created child contract

4) Attestation Request: After creating a child smart contract, the secondary artist would request the original owner to attest the newly created smart contract. The owner checks whether all the information has been correctly updated in the newly created smart contract and the smart contract meets the terms and conditions as agreed by both artists.

Figure 8 shows the log details of a successful attestation request sent by the secondary artist. The request includes the child smart contract address as well as the video details. At the end of a successful execution, a notification is sent out on the new attestation request as shown in Figure 8.


```
"string infor": "Comedy Chinese",
"bytes32 hash": "0x914a2bac6a231a389d48f1542e97f7d201131a91ff3fa839aa2cb6ac488b78

"string meta": "Nikon",
"address SCaddress": "0x1439818DD11823c45fFF01aF0Cd6c50934e27Ac0"

---

{
        "from": "0x692a70d2e424a56d2c6c27aa97d1a86395877b3a",
        "topic": "0xb9b72052a89e6a2f938dcb2d2d2868ff8fbb9343f1b6a9573626813554cea

        "event": "AttestationRequest",
        "args": {
                "0": "Address of child: ",
                "1": "0x1439818DD11823c45fFF01aF0Cd6c50934e27Ac0",
                "info": "Address of child: ",
                "Scaddress": "0x1439818DD11823c45fFF01aF0Cd6c50934e27Ac0",
                "length": 2
        }
}
```


Fig. 8: Logs showing an attestation request made by a secondary artist

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905689, IEEE Access

5) Attestation Granting: A request sent by the secondary artist requires the original owner to grant or deny attesting the newly created smart contract. This functionality can only be done by the original artist. The attestation is granted through a boolean which holds the values $true$ or $false$ based on the owner's attestation result. Figure 9 shows the logs generated after a successful attestation of a secondary artist smart contract by the original artist.

Fig. 9: A successful attestation granted to the smart contract of a secondary artist

V. Evaluation

In this section, we give a brief cost and security analysis of the proposed blockchain-based solution for providing proof of authenticity for digital content.

A. Cost Analysis

Every transaction performed on the blockchain network costs Gas which is effectively paid in Ether tokens. Ether is used in our proposed solution to pay for the costs of each transaction since our execution is on the Ethereum blockchain. For each function executed on the blockchain network, there are transaction and execution gas costs. The execution cost is effectively the cost of the actual execution of the function code handling the translation on the blockchain network. It includes the cost of the internal storage in the smart contract as well as any manipulation with the state. Moreover, the transaction cost includes other factors related to the deployment of the contract and sending the data to the blockchain network [28].

Table I shows the gas costs of the functions in the smart contract as well as their price in US Dollars. The gas price used in Table I is the average gas price on Nov 11th, 2018 which is 2.8 Gwei according to the ETH Gas Station [29].

The functions in Table I are either executed by the Original Artist (OA) or the Secondary Artist (SA) as seen in the Function Caller column of the table. The overall cost of the functions is minimal as all of them are less than $0.1 The operation that costs the least is the $AttestSC$ function. This is because it does not change a lot in the states of the variables in the smart contract. On the other hand, it can be seen that the $GrandAttestation$ function costs the most. This is also due to the fact that the function is considerably changing the state of the smart contract. Here, if the attestation is granted all the video details are updated in the original smart contract. This allows the original smart contract to keep track of all the videos of the child contracts. Consequently, the costs in our smart contract operations are proportional to the changes in the state of the smart contract.

TABLE I: Gas Costs of the Smart Contract Functions

Function Caller	Function Name	Transaction Gas	Execution Gas	Cost USD($)
SA	$RequestPermission$	72250	48802	0.042
OA	$GrantPermission$	109423	84375	0.064
SA	$UpdateProvenance$	129716	102172	0.076
SA	$AttestSC$	36220	8612	0.021
OA	$GrantAttestation$	163449	135649	0.095

B. Security Analysis

This section gives a brief security analysis on how our blockchain-based solution ensure key security goals such as integrity, accountability, authorization, availability and non-repudiation. We also discuss how our solution is resilient and is secure against popular attacks as those of Man In the Middle (MITM), replay and Distributed Denial of Service (DDoS) attacks.

- **Integrity:** It is important that all transaction history as well as the provenance data available for the users to track and trace a video to its origin are tamper proof. Our solution ensures the integrity of all the events and logs including relevant traceability provenance data are all stored in the immutable blockchain infrastructure. Moreover, a videos integrity is also maintained well by storing it on the IPFS distributed servers and only storing the hash in the smart contracts. Any change to the video will lead to a new hash that will not match the hash in the smart contract. Consequently, the video content on the blockchain is tamper proof as well as the reputation of the creator since the reviews are also IPFS based and cannot be altered.
- **Accountability:** Every function call in the smart contract executed by a caller on the blockchain is traced back to the Ethereum address of the caller. Therefore, every participating entity is accountable for its actions on the ledger.
- **Non-repudiation:** All transactions taking place on the blockchain network are cryptographically signed by the initiator. Hence, no one can deny their own actions as everything is saved in the tamper-proof logs.
- **Authorization:** In our smart contract code, every function can only be executed by a certain entity. This is done using modifiers, where a requirement is placed before the execution of a function code. Only if the Ethereum address matches the authorized executers Ethereum address, the function will be executed. Also, the design of the solution only allows a smart contract to trace back to a parent contract if the attestation is given by the original

creator. Hence, authorization from the original creator is required to finalize the on-chain provenance data update.

- **Availability:** The participating entities can always access the smart contracts once deployed to the blockchain network. All the logs as well as on-chain provenance data are accessible and made available to all through the ledger. Due to the decentralized nature of the blockchain and the global placement and distribution of the tens of thousands of mining nodes, the blockchain network is protected against **DoS and DDoS** attacks. The information stored on the ledger is saved in a distributed and decentralized way and is not subject to hacking, compromise or being a single point of failure. Tamper-proof records are replicated and available with all mining nodes.
- **MITM and Replay Attacks:** By design, our solution inherits the security features of the blockchain technology. Therefore, every transaction that gets executed on the chain is cryptographically signed by each participant's private key. Also each transaction has a unique timestamp and id. Hence, if an intruder tries to manipulate the content or modify it in any way, they cannot sign it without the legitimate private key, and therefore the transaction will be invalidated and discarded by the mining nodes. Duplicate transactions will also be discarded by the mining nodes. This makes the solution secure against replay and MITM attacks.
- **Impersonation and Sybil Attacks:** In impersonation, the attacker tires to masquerade as a legitimate user to get system authorization and access. In Sybil attacks, the attacker assumes many illegitimate and fake identities with the purpose of gaining more control and influence within a community of users. By design, blockchain prevents both attacks by having the identity of each participant/artist to be associated with a unique private key that is known only to the user. In our proposed system, the ENS service (which is a decentralized Etherem-based naming and registry services) has a record of all identities with the associate public key of all artists with identity attributes that include name, company and profile. An adversary will not be able to exercise any transaction without having the private key associated with the public key stored in the immutable ENS system.

VI. Conclusion

In this paper, we have presented a blockchain-based solution for proof of authenticity of digital videos in which a secure and trusted traceability to the original video creator or source can be established, in a decentralized manner. Our solution makes use of a decentralized storage system IPFS, Ethereum name service, and decentralized reputation system. Our proposed solution framework, system design, algorithms, sequence diagrams, and implementation and testing details are generic enough and can be applied to other types of digital content such as audios, photos, images, and manuscripts. Our solution can help combat deepfake videos and audios by helping users to determine if a video or digital content is traceable to a trusted and reputable source. If a video or digital content is not traceable, then the digital content cannot be trusted. Our smart contract-based solution provides a trusted way for secondary artists to request permission from the original artist to copy and edit videos. The full code of the smart contract has been made available at Github. Key features and functionality of the smart contract have been properly tested. We discussed how our solution meets security requirements, and is resilient against commonly known security attacks. We estimated the operational cost in terms of Ether and Gas when deploying the smart contract on the real Ethereum network. The cost estimate is minimal and is always under 0.095USD per transaction. As a future work, we are in the process of developing front-end DApps for users to automate the establishment of proof of authenticity of published videos. Also we plan to develop a pluggable DApp component to provide traceability and establish authenticity when playing or displaying videos within a web browser. Also work is underway for designing and implementing a fully functional and operational decentralized reputation system.

References

[1] When seeing is no longer believing: Inside the Pentagon's race against deepfake videos. January 2019. [Online]. Available: http://edition.cnn.com/interactive/2019/01/business/pentagons-race-against-deepfakes/

[2] Lawmakers warn of 'deepfake' videos ahead of 2020 elections. January 28, 2019. [Online]. Available: https://edition.cnn.com/2019/01/28/tech/deepfake-lawmakers/index.html

[3] How faking videos became easy and why that's so scary. [Online]. Available: http://fortune.com/2018/09/11/deep-fakes-obama-video/

[4] Are deepfakes the new fakenews? [Online]. Available: https://www.mediaupdate.co.za/media/144611/are-deepfakes-the-new-fakenews

[5] D. Stover, "Garlin gilchrist: Fighting fake news and the information apocalypse," *Bulletin of the Atomic Scientists*, vol. 74, no. 4, pp. 283–288, 2018.

[6] L. Floridi, "Artificial intelligence, deepfakes and a future of ectypes," *Philosophy & Technology*, vol. 31, no. 3, pp. 317–321, 2018.

[7] What counts as an artworks proof of authenticity. [Online]. Available: https://www.artsy.net/article/artsy-specialist-counts-artworks-proof-authenticity

[8] Is your certificate of authenticity worth the paper it's printed on? [Online]. Available: https://www.artbusiness.com/certaut.html

[9] Art provenance: What it is and how to verify it. [Online]. Available: https://www.artbusiness.com/provwarn.html

[10] K. Toyoda, P. T. Mathiopoulos, I. Sasase, and T. Ohtsuki, "A novel blockchain-based product ownership management system (poms) for anti-counterfeits in the post supply chain," *IEEE Access*, vol. 5, pp. 17 465–17 477, 2017.

[11] P. Treleaven, R. G. Brown, and D. Yang, "Blockchain technology in finance," *Computer*, no. 9, pp. 14–17, 2017.

[12] F. Tian, "A supply chain traceability system for food safety based on haccp, blockchain & internet of things," in *Service Systems and Service Management (ICSSSM), 2017 International Conference on*. IEEE, 2017, pp. 1–6.

[13] R. AlTawy, M. ElSheikh, A. M. Youssef, and G. Gong, "Lelantos: A blockchain-based anonymous physical delivery system," Cryptology ePrint Archive, Report 2017/465, 2017. http://eprint. iacr. org/2017/465, Tech. Rep.

[14] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *Open and Big Data (OBD), International Conference on*. IEEE, 2016, pp. 25–30.

[15] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[16] S. Singh and N. Singh, "Blockchain: Future of financial and cyber security," in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, Dec 2016, pp. 463–467.

[17] K. Biswas and V. Muthukkumarasamy, "Securing smart cities using blockchain technology," in *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Dec 2016, pp. 1392–1393.
[18] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[19] Ipfs is the distributed web. [Online]. Available: https://ipfs.io/
[20] Y. Li and S. Lyu, "Exposing deepfake videos by detecting face warping artifacts," *arXiv preprint arXiv:1811.00656*, 2018.
[21] Fight AI with AI! code taught to finger naughty deepfake vids made by machine-learning algos. [Online]. Available: https://www.theregister.co.uk/2018/11/06/fight_ai_deepfakes/
[22] Deepfake-busting apps can spot even a single pixel out of place. [Online]. Available: https://www.technologyreview.com/s/612357/deepfake-busting-apps-can-spot-even-a-single-pixel-out-of-place/
[23] Authenticity verification of user generated video files. [Online]. Available: https://prover.io/#intro
[24] B. Gipp, J. Kosti, and C. Breitinger, "Securing video integrity using decentralized trusted timestamping on the bitcoin blockchain." in *MCIS*, 2016, p. 51.
[25] Originalmy products. [Online]. Available: https://originalmy.com/products
[26] Ethereum name service. [Online]. Available: https://ens.domains/
[27] The civil white paper. [Online]. Available: https://civil.co/white-paper/
[28] Optimizing your solidity contracts gas usage. [Online]. Available: https://medium.com/coinmonks/optimizing-your-solidity-contracts-gas-usage-9d65334db6c7
[29] Eth gas station. [Online]. Available: https://ethgasstation.info/