Blockchain-based P2P multimedia content distribution using collusion-resistant fingerprinting

Amna Qureshi and David Megías
Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya (UOC),
Center for Cybersecurity Research of Catalonia (CYBERCAT)
Av. Carl Friedrich Gauss, 5, 08860, Castelldefels, Spain
E-mail: {aqureshi,dmegias}@uoc.edu

***Abstract*—Due to the popularization of low-cost broadband Internet access, the amount of the digital data that is illegally redistributed is growing, making content creators and owners lose their income. Fingerprinting, a watermarking-based technology for embedding buyer identifications in legally distributed contents, has emerged as a promising approach to fight illegal redistribution. On the other hand, the blockchain technology is also shaping up to handle the challenge of digital copyright protection. With blockchain, media producers can authorize and manage their copyrights on a public ledger. In this paper, we present a blockchain-based distribution system which blends different technologies (collusion-resistant fingerprinting, perceptual hash functions, and a peer-to-peer file distribution network) to provide copyright protection, collusion resistance, atomic payment, piracy tracing, transparency, proof-of-delivery, revocable privacy (to a buyer), and dispute resolution. The paper also analyzes several security and privacy compromising attacks and countermeasures.**

***Keywords*—Collusion-resistant fingerprinting; Blockchain; Privacy; Security; Smart contracts; Cryptography.**

I. Introduction

In recent years, the prosperity of digital and information technologies has opened limitless channels for distribution of multimedia content. In the past, the content distribution was limited to tightly controlled broadcasts or the sale of analog media, but, with the digital revolution, the Internet has emerged as an efficient content distribution channel. Examples of content distribution include, but are not limited to, bulk data transfer, streaming continuous media, shared data applications, and interactive gaming. The content providers need to distribute their respective content efficiently to end users.

The cost-effective multimedia content distribution over peer-to-peer (P2P) networks has become very popular in recent years. However, from P2P file-sharing services to streaming continuous media, copyrights of content producers are not respected. These systems have been used on a large scale for illegal purposes such as downloading of copyrighted materials. Such practices have a detrimental effect on multimedia producers who are being robbed of their royalties because of unpaid and unauthorized downloads. Lack of copyright protection mechanisms within these systems prevent content providers from adopting this technology despite its advantages such as scalability, cost-efficiency, fault tolerance, etc. Also, traceability of copyright violators within a P2P system is a tedious task. Thus, content protection mechanisms (e.g., digital rights management (DRM), watermarking, etc.) are required in P2P-based content distribution systems to ensure fair and legal usage of copyrighted multimedia content. A content protection technique, digital fingerprinting, addresses the problems of both copyright protection and traitor tracing. It involves the generation of a fingerprint (a user specific identification mark), the embedding operation and traceability from redistributed copies. A few examples of P2P systems based on digital fingerprinting are [1], [2], [3], [4], [5], [6]. Though the distribution systems based on digital fingerprinting address the problems of copyright protection and traceability, these is no effective mechanism to provide proof-of-delivery of the digital content to the end users (buyers). Also, the buyers are required to deposit the payment of the content before its delivery. This proof-of-delivery is necessary that ensures the buyer that after the payment has been made, he/she receives untampered and as requested digital content. Also, within these content distribution systems, income transmission from the buyer to the content provider lacks transparency. Often these systems use centralized trusted third parties for payment mechanism, which are vulnerable to single point-of-failure, compromise and hacking attacks.

A decentralized and transparent content distribution system with proof-of-delivery is possible using blockchain technology [7], which is known for being immutable, traceable, and tamper-proof with a distributed ledger. The blockchain technology is relevant to anything that requires a transaction verification leading to authenticity, integrity and trust. This technology has contributed to several solutions, not only in finance, but also in health care, supply-chain management, and intrusion detection. The key features of blockchain technology –i.e., decentralization, traceability, scalability and transparency– provide novel ideas for copyright protection and traceability. Using blockchain, the results of copyright generation and transaction recording, transmission and storage are trustworthy, and the recorded information cannot be tampered once it is generated. Recently, many systems have been proposed to provide copyright protection using blockchain, but these are based on DRM approach. The combination of copyright protection schemes based on watermarking or fingerprinting, and the blockchain has not received much attention from the researchers. In [8], blockchain is used to store watermark securely and provides timestamp authentication

for multiple watermarks (multiple copyrights) to confirm the creation order. The authors in [9] have proposed a scheme for the protection of network copyright transaction of images. The realization structure of copyright market transaction based on blockchain technology is introduced, and a specific business model of image copyright transaction is described. In [10], the authors have designed and implemented an Ethereum application, BMCProtector, which is based on blockchain and smart contract technologies, to protect music copyright and rights of copyright owners. To prevent piracy, encryption and digital watermarking are used.

In the above mentioned schemes and a few others [11], [12], either digital watermarking or DRM is used as a tool to provide copyright protection. In this paper, we propose a content distribution system that combines collusion-resistant fingerprinting and blockchain technologies to provide copyright protection, collusion resistance, piracy tracing, transparency, proof-of-delivery, and revocable privacy (to a buyer). Various algorithms are designed to be incorporated within the Ethereum [13] smart contract to execute secure content delivery from the content owner to the buyer, automate payment between the content provider and the buyer in Ether cryptocurrency without the involvement of any centralized trusted third party (for payment), settle payment between the content provider and the buyer to end the transaction, trace copyright violator in case of illegal copy redistribution, and resolve disputes if arisen. Also, in the proposed system, the Ethereum blockchain is integrated with the InterPlanetary File System (IPFS) [14], a P2P file system that is run by multiple nodes, storing files submitted to it. After the generation of copyrighted content, the content provider uploads it to IPFS, which assigns a unique hash code for it in the blockchain.

The rest of the paper is organized as follows. Section II defines the basic building blocks of the proposed system. In Section III, the design and functionality of the proposed system is described in detail. Section IV describes the functionality of the smart contract. Section V analyzes the security and privacy properties of the system. Finally, in Section VI, we present the conclusions of this work.

II. Building Blocks

Collusion-resistance fingerprint protocol, Quantization Index Modulation (QIM) watermarking, a homomorphic encryption scheme, blockchain and smart contracts, IPFS, and hash functions are the basic technologies that are combined to build the proposed blockchain-based content distribution system.

A. Collusion-resistant Secure Codes

Nuida et al.'s codes [15] are used in the proposed system to provide collusion-resistant against colluders. The algorithm for the fingerprint generation takes parameters ϵ, N and c as inputs, and outputs a collection $F = (f_1 \dots f_N)$ of binary codewords f_i of size m and a secret vector p. The codeword f_i is meant to be embedded into a content of a buyer i. The details of Nuida et al.'s codes construction and traitor-tracing algorithm can be found in [15].

B. Quantization Index Modulation Watermarking

QIM is a watermark embedding technique that embeds a fingerprint bit f by quantizing selected values (e.g. DWT coefficients) by choosing between a quantizer with even or odd values, depending on the binary value of f. It is important to consider an optimal selection of the embedding quantizer step size δ and a scaling factor so that the best tradeoff between robustness and minimum quality degradation can automatically be achieved. In our system, a blind, robust and secure QIM-based watermarking technique [16] is employed to embed Nuida et al.'s codewords into the content.

C. Homomorphic Encryption

Homomorphic encryption systems allow operations to be performed on encrypted data without compromising the encryption. These schemes are used in asymmetric fingerprinting protocols to provide buyer frameproofness against a dishonest content owner. In our system, we have used a Paillier cryptosystem [17], which is homomorphic with respect to the addition operation, to insert an encrypted Nuida et al.'s fingerprint codeword into the encrypted multimedia content.

D. Blockchain and Smart Contracts

Blockchain is an open distributed ledger that records all transactional details referred as blocks. Each record or block is time-stamped and linked to a previous block using a cryptographic hash. Data written to a blockchain is immutable and, thus, provides an auditable record of events and logs that cannot be modified or deleted. Also, an exact copy of blockchain is maintained in a large number of independent locations and, therefore, there is no central point of failure. It operates in a peer-to-peer fashion by allowing transactions to be verified without the need of supervision by any trusted third party. Instead, multiple nodes are used to form a consensus on whether a transaction is valid or not. Invalid transactions are not acknowledged and rejected. A transaction will only be added to a block when it obtains the majority consensus after verification by all nodes.

A smart contract [7] is a self-executing program that runs on a blockchain with a unique address. A smart contract allows transactions to be carried out between different entities without the need of a central authority, and enables the code to execute autonomously upon meeting specified conditions. It can store information as internal state variables and define custom functions to manipulate or update its state. The operations in a smart contract are published as transactions. These operations are deterministic and verifiable by transaction nodes to ensure their validity. Each smart contract creates events and logs that help in tracing. Etherium, a public, open-source, immutable blockchain, allows execution of smart contracts without any third party interference and offers Turning-complete languages with more expressive expressions. A smart contract consists of the following entities:

- Functions: Methods that are used to have the tasks done by the involved entities.

- Modifiers: Used to set and modify state variables based on meeting certain requirements of the contract.
- Events: Raised by the execution of any function call, they notify the involved parties about updates regarding transactions taking place in the chain.
- Variables: Values that change depending on the conditions. A variable can store specified data type depending on the contract conditions.

In this system, we make use of security, reliability, and pseudo-anonymity features of blockchain to keep all records of the transactions between the content owner and the buyer for tamper-proof verifiable integrity of the content. We use smart contracts on an Etherium blockchain to ensure the reliability of copyright transactions involving an asymmetric fingerprinting protocol between the content owner and the buyer.

E. InterPlanetary File System (IPFS)

IPFS is a high-throughput and content-addressed peer-to-peer file storage system, in which each peer stores a collection of hashed files. A user who wants to retrieve any of these files can access a friendly abstraction layer, where he/she inputs the hash of the file that he/she wants. IPFS then searches through the peers and supplies the user with the required file. Duplicate entries (files with the same content) are removed across the file system and version history is tracked for every file. IPFS enables high volume data distribution with high efficiency and persistent availability of content secured and verified by cryptographic hashing. Data is verified with its checksum. Hence, if the hash changes, the IPFS will know the data is tampered. IPFS is interoperable with smart contracts, and thus, can add reliable and low-cost storage capacity to a blockchain ecosystem. In the proposed system, IPFS is used as an external distributed storage medium that stores fingerprinted and non-fingerprinted multimedia files. Also, IPFS is used within a blockchain transaction to place immutable and permanent content links in the blockchain.

F. Hash Functions

Conventional hash functions are not suitable for the multimedia domain because they are too sensitive to data modification. If the data is changed by one bit, a cryptographic hash value will change completely. A solution to this shortcoming is a perceptual hash function that is based on extracting robust and distinctive features from multimedia data. These features are resistant to incidental distortion, such as compression, noise addition or other signal processing operations, and make perceptual hashing a promising tool for multimedia identification and authentication. In the proposed system, a perceptual hash function [18] is used to calculate the hash value of the multimedia file as its ID number, which is recorded in the blockchain to be used later in the PaymentSettlement$(\cdot)$ function of the proposed smart contract to verify the authenticity of the downloaded content. A multihash function [14] is also used in the proposed system to provide content addressing in IPFS. It is a function for differentiating outputs from various well-established hash functions, addressing size and encoding considerations. Additionally, SHA-1 is used to provide data integrity and authenticity of the messages exchanged between different parties.

III. Proposed System

This section describes the design and functionality of the proposed system. In Section III-A, we describe the role of each entity. Section III-B defines the functionality requirements and the security assumptions. An attack model is described for the system in Section III-C.

A. System Entities

The proposed system consists of six basic entities whose functionality is defined as follows:

- Content owner: A content owner *CO* is an entity that owns a digital content and is interested in selling the copyrighted content to interested buyers by registering the content on a blockchain through the smart contract. It is involved in seven functions of the smart contract: content registration and agreement, fingerprint generation, base and supplementary files generation and distribution, traceability and arbitration.
- Buyer: A buyer B is an entity who is interested in buying content(s) from the content owner by requesting it through the smart contract. It is involved in the following functions of a smart contract: content query and agreement, Ether deposit, acquisition of base and supplementary files from IPFS, payment settlement, and a dispute resolution, in case he/she is found guilty of copyright violation.
- Monitor: A monitor (*MO*) is a trusted party which is responsible for the generation of Nuida et al.'s fingerprinting codes. The existence of *MO* ensures that the generated fingerprints are not revealed to *CO* and the buyer, thus resolving the problems of customer's rights and non-repudiation. The hashes of the fingerprints and encrypted fingerprints along with associated details (transaction ID, timestamp, etc.) are registered to a blockchain to be later used in the traceability phase. Also, *MO* provides traceability of a copyright violator by executing a Nuida et al.'s [15] traitor-tracing algorithm in case of a piracy claim by *CO*.
- Arbitrator: An arbitrator is assumed to be a trusted party which is involved in dispute resolution process of the smart contract to resolve disputes between *CO* and the buyer. Based on the results, a refund may take place.
- IPFS: IPFS is a file server that stores the base and supplementary files generated by *CO*. An interested buyer, upon depositing the content payment, can download the files against the provided hashes of the base and supplementary files from IPFS through the smart contract. Also, IPFS notifies the smart contract when the files are downloaded successfully by the buyer.
- Certification Authority: The Certification Authority (*CA*) is a trusted entity that is responsible for generating cryptographic key pairs, and issuing these key pairs to the

requesting users upon successful authentication. For each new transaction, the *CA* generates and issues temporary key pairs to the authenticated system users.

B. Design Requirements and System Assumptions

In this section, the design requirements, security assumptions and threat model of the proposed system are described.

1) Design Requirements: Following are the design requirements of the system:

- Non-repudiation: The buyer accused of redistribution of an unauthorized copy should not be able to claim that the copy was created by *CO*.
- Buyer-frameproofness: *CO* should not be able to frame an honest buyer for illegal redistribution.
- Traceability: *CO* should be able to trace and identify an illegal redistributor in case of finding a pirated copy with the help of trusted parties (*MO* and *CA*).
- Collusion-resistance: The scheme should be collusion resistant against a given number of colluders c as specified by Nuida et al.'s codes [15].
- Dispute resolution: The arbitrator, with the help of trusted third parties, should be able to resolve the disputes between two conflicting parties (*CO* and B).
- Watermarking properties: The embedding process should be blind and the embedded fingerprint should be imperceptible and robust against common signal processing attacks.
- Privacy: The real identity of the buyer should remain anonymous during content purchase unless he/she is proven guilty of copyright violation.
- Secure transfer of files: Transfer of the base and supplementary files from *CO* to IPFS, and IPFS to the buyer must be secure.
- Efficiency: The data expands on conversion from a plaintext to an encrypted representation of signals due to the use of an additive homomorphic cryptosystem. The homomorphic encryption should be performed in such a way that the size of the encrypted base file remains small.
- Fairness: At the end of the smart contract, the buyer obtains a valid content and *CO* obtains his/her payment. In case of a dispute, an Arbitrator should resolve it and settle the payment.
- Revocable agreement: If the buyer is found guilty of copyright violation, the agreement between *CO* and the buyer should be revoked. Also, the buyer should be punished by making a deduction of cryptocurrency equal to the cost of that content from his/her account.

2) System Assumptions: The underlying design and security assumptions of our scheme are described as follows:

- There are six major players involved: content owner, buyer, monitor, Arbitrator, IPFS and *CA*.
- *CA* is a trusted entity. Hence, *CA* will not form a coalition with any other party (*CO*, MO or Arbitrator) to break any user's privacy.
- *CA* is the only party who knows the real identity and the pseudonyms associated with it. *CA* keeps track of all the pseudo-identities to be sure that they remain unique, and also to revoke an identity of a malicious entity.
- *CO* and the buyer do not trust each other but they both trust *MO* and Arbitrator. In case of traitor tracing and arbitration, it is expected that neither *MO* nor Arbitrator forms a coalition with any other party to frame a buyer.
- A secret key (sk) used by *CO* to select fingerprint embedding positions remains constant for multimedia contents with the same perceptual hashes (i.e. different versions of the same content that are distributed to different buyers). The encrypted sk is recorded on the blockchain by *CO*. The same content for different buyers must have the same encrypted sk.
- Each entity has a pair of keys (of length 1024-bits) for encryption/decryption of data, and signing/verification of signatures.
- The reconstruction of the original file from the base and supplementary files is performed at the buyer's end without his/her assistance. The base file cannot be shared or redistributed.
- The proposed system makes use of Ethereum smart contract, and payments are made in Ether (the cryptocurrency token for Etherium blockchain). The payments are to be made to the smart contract that distributes it fairly to both *CO* and the buyer.
- Each entity has an Ethereum contract with an Ethereum Address (EA) and an asymmetric key pair.
- The functions of a smart contract can only be executed by authorized entities (entities with authorized Ethereum addresses). The buyer can register his/her EA so as to obtain access to specific functions.
- The generation of a fingerprint, the creation of the base and supplementary files, content downloading, the file reconstruction at buyer's end, and traitor tracing are performed off-chain, whereas all other functions of the smart contract are performed on chain.
- IPFS hashes of base and supplementary files are stored within the smart contract.
- A buyer is provided with the IPFS hashes of the base and supplementary files, and a unique token per transaction to download the content from the IPFS network.
- Cryptographic primitives and constructions used in the system are secure and verifiable.
- The system assumes a TLS channel between a buyer and IPFS to establish a secure channel to download the content.
- At the end of a successful content transfer, the buyer receives half of his/her initial deposit, whereas the other half is paid to *CO*.

C. Attack Model

This section describes the main attacks that may be aimed to break either the security or the privacy properties of the proposed system.

1) Framing Attack: When the fingerprint is inserted solely by *CO*, he/she may benefit from a framing attack on an

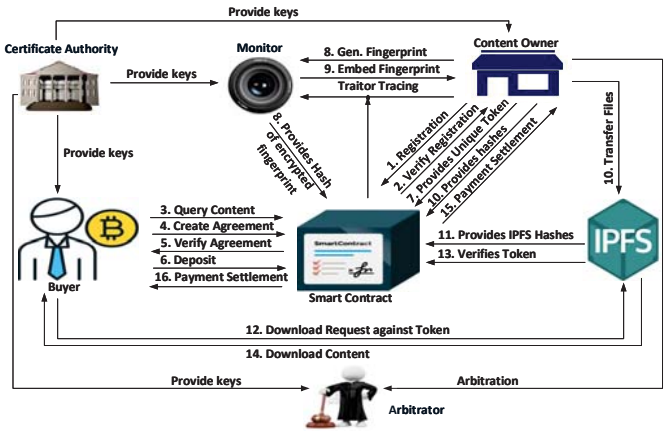

Fig. 1. Overview of the system

innocent buyer. This attack is successful if *CO* is able to prove to the Arbitrator that illegal copies of the marked content belongs to a particular buyer even though that buyer has not bought this content, or has bought this content but did not redistribute copies of it illegally.

2) Attacks on a Smart Contract: We consider the following possible attacks on a smart contract by an adversary:

- The adversary may attempt to alter the messages exchanged between the participating entities on the Ethereum blockchain.
- The adversary may eavesdrop on off-chain communication between different entities such as a buyer and IPFS or *CO* and *MO*.
- A malicious entity may attempt to mimic a buyer's EA to obtain the fingerprinted content.
- The adversary may attempt to call a specific function of the smart contract to gain access to a token or any other private information.
- An attacker may attempt to prevent the publication of a valid transaction in the blockchain.

The security against these attacks is discussed in Section V.

IV. Overview of the System

Fig. 1 illustrates the general architecture of the proposed content distribution system highlighting its main entities (*CO*, buyer, *MO*, Arbitrator IPFS, and *CA*) who interact with the smart contract. The proposed system makes use of an Ethereum smart contract, which consists of functions and events to execute secure content distribution between *CO* and the buyer.

Fig. 2 shows the possible sequence flow of transactions occurring from different functions among system entities. It shows the complete sequence of events beginning with the content registration and ending with the dispute resolution. A function, Registration(·), is initiated by *CO* to register his/her content on blockchain. Upon its registration, *CO* is notified about it with an event. *CO* verifies and attests the contract by examining its code and hash value. Upon successful verification, the content becomes available for the buyers to purchase it. Only the registered users of the Ethereum blockchain are permitted to query the content by executing QueryContent(·) function.

If interested, the buyer initiates CreateAgreement(·) function to create an agreement between *CO* and him/her. The created agreement (AGR) may include details such as title, publishing year, digital rights, content signature, *CO*'s signature, public key and EA, buyer's temporary public key (K^*_{pB}) and pseudo-identity, and content price. Once AGR is created, an event notifies *CO* about it. The buyer verifies AGR by either agreeing or disagreeing to its terms and conditions. Upon verification, the buyer initiates the DepositEther(·) function to pay for the content. The buyer deposits twice the amount of the total price to ensure secure transaction and payment settlement. If the buyer fails to deposit this much amount, the transaction aborts. On receiving the deposit, the smart contract informs *CO* with an event, and then *CO* provides an encrypted token for a buyer (a unique token encrypted with K^*_{pB}) to download the content by calling the ProvideToken(·) function, following which *CO* initiates an off-chain communication with *MO*, who is responsible for providing an encrypted collusion-resistant fingerprint to *CO* for embedding it into the content. *MO* registers hash of both fingerprint and encrypted fingerprint along with other details on chain by calling ProvideEncFingerprint(·) function. Meanwhile, *CO* generates two files: a fingerprinted base file and a supplementary file, which are then transferred to the IPFS network (since storing the multimedia files on chain would be expensive, these files are stored here). IPFS registers hash of both files on the chain by calling ProvideIPFSHashes(·) function. These hashes are provided to the buyer for accessing the files from IPFS. Upon receiving the hash values, the buyer requests for the content transfer off-chain from IPFS against the provided encrypted token. IPFS verifies the encrypted token stored on chain, and upon successful verification, allows the buyer to download the content off-chain against IPFS hashes. Once the buyer downloads the files, IPFS sends a confirmation to the buyer off-chain. The buyer reconstructs the file by combining the base and supplementary files, and computes a perceptual hash of the content to end the transaction and settle the payment. VerifyPerceptualHash(·) function is called by the buyer to evaluate if the computed hash and the hash stored on chain are equal. If the returned result is true, then the smart contract settles the payments to the involved entities, i.e., pay half of the deposit (price of the content) to *CO*, and refund half of it to the buyer. If the returned result is false, the Arbitrator is requested to resolve the dispute between *CO* and the buyer.

At anytime after a successful transaction, in case an illegal redistributed copy is found by *CO*, the Traceability(·) and Arbitration(·) functions are called.

A. Setup

Ethereum blockchain provides pseudo-anonymous accounts, i.e., public addresses composed of random hash values for users to perform transactions. It is globally accessible to anyone with Internet access and allows users to generate any number of blockchain accounts to minimize the iden-

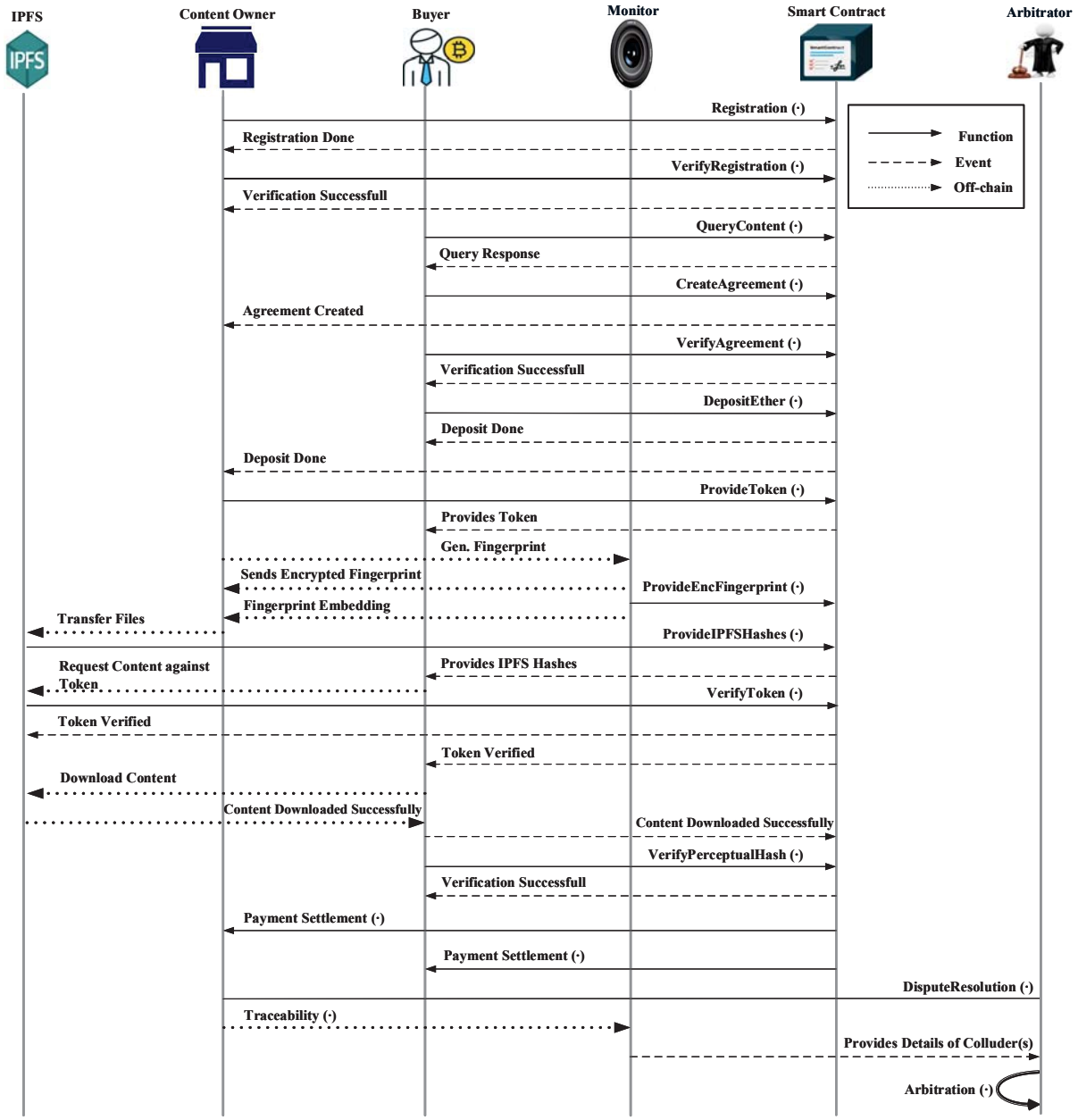

Fig. 2. Message flow of transactions

tifiability of account holders. However, the requirement of buyer's revocability in case he/she is found guilty of illegal redistribution implies the need for traceable user accounts. Thus, a mechanism is required that provides traceable identities while protecting the sensitive personal information on the blockchain. The proposed system employs public key cryptography to manage users' identities and provide confidentiality. *CA* is responsible to provide each entity a cryptographic public/private key pair of length 1024-bits. For each different transaction, the buyer generates a pseudo-identity with the help of the *CA*, who stores the user's real identity and the pseudo-identity (PI_B) in its database. For each new transaction, the buyer obtains a temporary key pair (K^*_{pB}, K^*_{sB}) from *CA*. The buyer records PI_B in the blockchain as his/her public identity (EA) for purchasing content from *CO*.

B. Registration

Fig. 3 illustrates the Registration(·) function of the smart contract. *CO* uses a cryptographic hash function to obtain the cryptographic hash value of this content as a content signature. Then, *CO* use perceptual hash function to calculate the perceptual hash value of this content. *CO* generates a secret embedding key (sk) depending on the perceptual hash of the content, and stores it in its database. All version of the same content (the same perceptual hashes) will use the same sk. *CO* encrypts sk with the Arbitrator's public key and records the

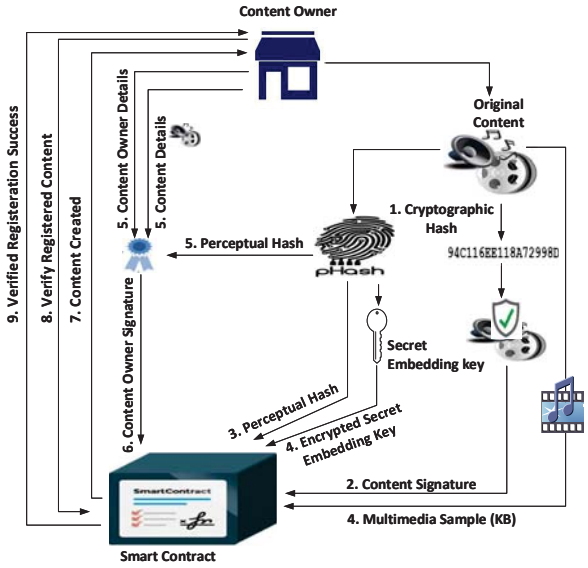

Fig. 3. Registration Function

encrypted sk on the blockchain to be later used in the payment settlement phase. A small sample of this content (audio or video) of a limited size (in KBs) is provided by *CO* to enable the buyers to check if the quality of the content will suffice to their needs of watching the video or listening the audio, or if any particular codecs are required at the buyers' end to play the content. *CO* generates his/her digital signature using his/her personal information, the content's metadata (such as title, type, publishing year etc.) and the perceptual hash value of the content. Using the content signature, the sample, the perceptual hash, the encrypted sk, and his/her own digital signature, *CO* initiates the Registration$(\cdot)$ function of the smart contract to record this transaction in the blockchain. Once the transaction is successfully recorded on chain, an event triggered by the smart contract notifies about it to *CO*, who can then verify the SHA hash of the content.

C. Agreement

The buyers searches in the blockchain for a content he/she is interested in. On finding the desired content, the buyer initiates an Agreement$(\cdot)$ function with the smart contract to setup an agreement (AGR) with *CO*. This AGR uniquely binds the content to this particular transaction between *CO* and the buyer. In this transaction, the buyer uses PI_B as EA to preserve his/her privacy. The smart contract upon receiving the request, creates AGR between *CO* and the buyer, stating the rights and obligations of both parties, credentials of *CO* (EA) and the buyer (EA), public keys of both *CO* and the buyer, and the specific details of the content (title, price, etc.). The buyer and *CO* are notified with an event about the creation of AGR. *CO* generates a transaction ID (TID) for keeping a record of AGR between him/her and the buyer. Then, the buyer verifies the terms and conditions stated in AGR. Upon agreeing, he/she is allowed to deposit Ether twice the actual cost of the content to make a purchase, else the transaction is terminated. The interested buyer is required to deposit twice the actual price to incentivize honesty. Upon receiving the deposit, the smart contract notifies both the buyer and *CO*, who then creates a unique token for the buyer that has a validity depending on the type of the requested content. This unique token is generated by *CO* to allow only the privileged party to download the content. Since it is registered on chain, it is possible that it can be viewed by anyone. Therefore, it is necessary to record the encrypted token on chain, thus, *CO* encrypts the token with K^*_{pB} and records it on the smart contract making the encrypted token available to the buyer to enable him/her to download the content.

D. Generation of the Base and Supplementary Files

Once *CO* is notified about deposit of Ether against AGR, it initiates a function that generates two files –a base file (*BF*) and a supplementary file (*SF*)– based on the design proposed in [2]. *BF* is designed to have a small size than the original file and contains the most important information. Without this information, the file reconstructed by the buyer would be unusable. *SF* is absolutely useless without the corresponding *BF*. *SF* can be shared or redistributed among IPFS users without any condition, whereas, *BF* cannot be shared or redistributed by the buyers.

1) Fingerprint Generation: *CO* sends a request for a fingerprint to *MO* by sending TID, AGR, the signed AGR (signed using its own secret key), his/her public key, K^*_{pB}, PI_B, and the timestamp. *MO* validates the signatures of both *CO* and the buyer from *CA*. After successful verification, *MO* generates a Nuida et al.'s c-secure codeword f_i of length m against a TID sent by *MO*. The details of the Nuida et al.'s fingerprint generation algorithm can be found in [15]. Then, *MO* encrypts the fingerprint with K^*_{pB}, and sends the encrypted fingerprint, the signed encrypted fingerprint (signed using its secret key), the signed AGR, the public key of *CO*, the timestamp, and TID to *CO*. *MO* stores AGR, the signed AGR, the encrypted fingerprint, the hashes of both fingerprint and encrypted fingerprint, the public key of *CO*, K^*_{pB} and PI_B against TID at its end. *MO* interacts with the smart contract to record the hashes of fingerprint and the encrypted fingerprint, TID, the timestamp, AGR, the signed AGR, PI_B, K^*_{pB} and the public key of *CO* on the blockchain. Upon receiving the encrypted fingerprint from *MO*, *CO* records hash of encrypted fingerprint, the content signature, the perceptual hash, AGR, the signed AGR, TID, PI_B, K^*_{pB}, the timestamp, and his/her own public key on the blockchain.

2) Fingerprint Embedding: Figure 4 shows the generation of *BF* and *SF*. *CO* employs the Discrete Wavelet Transform (DWT) to split the content into approximation (low-frequency) and detail (high-frequency) coefficients. An approximation coefficient is then itself split into a second-level approximation and detail coefficients, and the process is repeated as many times as desired (levels of decomposition). In order to embed an encrypted fingerprint obtained from *MO* in the approximation coefficients to form *BF*, the additive homomorphic property of public-key cryptosystems is applied to encrypt approx-

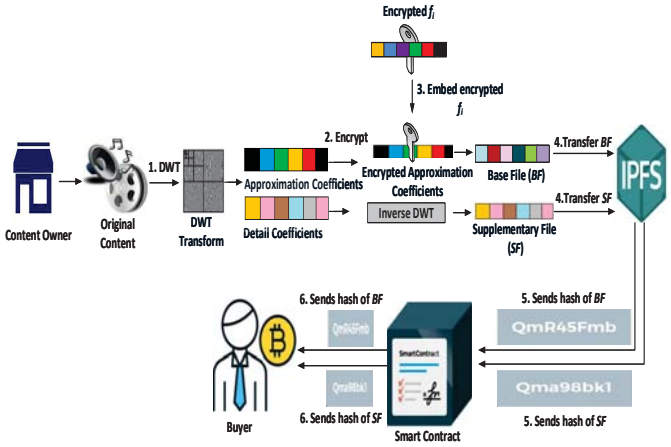

Fig. 4. Fingerprint Embedding

imation coefficients with K^*_{pB}. Since additive homomorphic cryptosystems cannot work on real-valued DWT coefficients, these approximation coefficients are quantized to integer values. In the quantization process, the approximation coefficients are quantized to the nearest even/odd integers depending on the value of quantization step size δ, that is a positive integer such that all the quantized coefficients are encrypted. Before quantization, *CO* selects the fingerprint embedding positions by using sk, which is also used to extract the fingerprint from the redistributed copies. The selected quantized coefficients are encrypted coefficient-by-coefficient with K^*_{pB}. To embed a single bit of a Nuida et al.'s codeword into one of the selected quantized and encrypted coefficient, SD-QIM watermarking technique [16] is performed. Also, *CO* encrypts the remaining scaled and quantized approximation coefficients that do not carry a fingerprint so as to hide these embedding positions. These approximation coefficients are encrypted in a block form with K^*_{pB} instead of encrypting individual individual coefficients to get a reduced *BF* size. All these approximation coefficients, i.e., the quantized and encrypted approximation coefficients and the embedded coefficients are recombined to constitute *BF*. On the other hand, an inverse L-level DWT is performed on the detail coefficients to obtain *SF*.

Upon generation of *BF* and *SF*, these files are automatically transferred to the IPFS network off-chain for buyers' downloads. Also, *CO* records content signature, perceptual hash, AGR, signed AGR, TID, encrypted fingerprint and its hash, encrypted sk, his/her own public key, K^*_{pB}, PI_B, hash of *BF* and *SF*, and timestamp on the blockchain. IPFS generates the hash of *BF* and *SF*, thus, indicating successful upload on the network. IPFS initiates ProvideIPFSHashes$(\cdot)$ function to record both IPFS hashes of *BF* and *SF* on the blockchain. Upon successful creation of this transaction, the smart contract notifies the buyer about the availability of the purchased content on IPFS by providing him/her these hashes.

E. Content Delivery

A TLS channel is assumed between a buyer and the IPFS network for secure transfer of messages (setting up of TLS channel is out of scope of this paper). Upon receiving the IPFS hashes of *BF* and *SF*, the buyer interacts with the IPFS network to download the content using the provided encrypted token. IPFS, upon receiving the demand from the buyer, initiates VerifyToken$(\cdot)$ function for verification of the received token. If the received token (from IPFS) and the token stored on the blockchain are the same, the smart contract sends a successful verification notification to IPFS and the buyer, who then calls the hashes from IPFS to download *BF* and *SF* off-chain. Upon successful download, IPFS notifies the buyer off-chain. Also, IPFS sends a notification with an event to the smart contract about the successful file transfer.

1) Payment Settlement: Once files are downloaded, the buyer performs decryption on the encrypted *BF* using K^*_{sB} and apply the inverse DWT to get a fingerprinted *BF*, which is then recombined with *SF* to obtain the complete content. The buyer uses the perceptual hash algorithm [18] to compute a perceptual hash of the generated content to end the transaction and settle the payment with the smart contract. The VerifyPerceptualHash$(\cdot)$ function within the smart contract is called by the buyer to compare the computed hash with the hash stored on chain. If the returned result is true, then the smart contract settles the payments to the involved entities, i.e., pay half of the deposit (total price of the content) to *CO*, and refunds half of it to the buyer, else DisputeResolution$(\cdot)$ function is initiated with the Arbitrator to resolve the dispute between *CO* and the buyer. Here there can be three possibilities: (1) file reconstruction error at the buyer's end; or (2) malicious act by *CO*; or (3) manipulation of fingerprinted content by the buyer.

In either case, the smart contract requests the Arbitrator to step in to resolve the dispute between *CO* and the buyer by providing him/her all the details regarding the content purchase against TID. The Arbitrator requests the buyer to send him/her the downloaded *BF*. Upon receiving the decrypted *BF*, the Arbitrator extracts the fingerprint by decomposing *BF* with the same wavelet basis used in the fingerprint insertion step. This gives the approximation coefficient matrix in which the fingerprint code is embedded. The Arbitrator retrieves encrypted sk against TID from the blockchain, decrypts it using its secret key, and extracts the code from secret embedding positions. The details of fingerprint extraction technique can be found in [2]. After the extraction of the fingerprint code, the Arbitrator computes the hash of the extracted code and compares it with the hash of the fingerprint code retrieved from the blockchain. If both hashes are equal, the Arbitrator notifies the buyer to try to download the content again from the IPFS network. Upon second attempt, if the buyer again complains about mismatch between perceptual hashes, the Arbitrator notifies the smart contract about a malicious *CO* and requests the refund of the buyer's deposit. In case both hashes are not equal, the Arbitrator notifies the smart contract to revoke AGR of *CO* with the dishonest buyer, and requests to punish him/her by deducting Ether equal to the total cost of the content from his/her account.

F. Traceability

Upon finding a pirated copy of the content, *CO* initiates Traceability$(\cdot)$ function with the help of *MO*. *CO* calculates a perceptual hash of the content and looks into its database against it to find the corresponding sk to extract the pirated code. *CO* then sends the pirated code to *MO*, who executes Nuida et al.'s codes tracing algorithm to identify the colluder(s). The details of the tracing algorithm can be found in [15]. This tracing algorithm outputs a user or a set of users with the highest score. *MO* sends the pseudo-identities of the colluder(s) to the Arbitrator, who then initiates the Arbitration$(\cdot)$ function. Also, the pseudo-identity of a colluder, *CO*'s signature and public key, TID, the content signature, AGR, the signed AGR and the timestamp are recorded by *MO* on the blockchain.

G. Arbitration

Upon receiving information from *MO*, the Arbitrator sends a request to *CA* to reveal the real identity of the colluder(s) by sending it the pseudo-identity of this entity. *CA* looks into its database for an entry against the provided pseudo-identity and sends the real identity to the Arbitrator, who notifies the smart contract to revoke AGR with the colluder(s), and punish him/her by deducting Ether equal to the total cost of the content from his/her account. In case *MO* did not find any colluder, it notifies the Arbitrator, who then requests the smart contract to refund the buyer's deposit and revoke AGR between dishonest *CO* and the buyer.

V. Security and Privacy Analysis

This section analyzes the privacy and security properties of the system according to the design requirements and the attack model presented in Sections III-B1 and III-C, respectively.

A. Content Owner's Security

The proposed system is secure and fair from the perspective of *CO* because a buyer has no idea about the embedded fingerprint, which is generated by *MO*, an entity trusted by both the buyer and *CO* (as described in Section III-B2). Thus, the buyer cannot accuse *MO* of collaborating with *CO* to frame him/her. Also, the buyer cannot claim that a pirated copy is created by *CO* since only he/she can decrypt the encrypted *BF* with his/her secret key. Moreover, the fingerprint is embedded into the selected positions of the content. Thus, the probability to find the exact locations of the embedded fingerprint is quiet low. Furthermore, a malicious buyer's claim of innocence can be rejected by the Arbitrator by using the hash of the fingerprint stored on the blockchain. Moreover, a traitor-tracing mechanism is proposed to unambiguously identify a copyright violator once a pirated copy is found.

B. Buyer Security

CO knows only about the encrypted fingerprint (encrypted with K^*_{pB}) and the encrypted *BF* and has no knowledge about K^*_{sB}. Therefore, *CO* does not know about the fingerprinted copy that the buyer obtains after decrypting the encrypted *BF*. Furthermore, the transaction recorded on chain by *MO* containing the hashes of the fingerprint and the encrypted fingerprint, AGR, the signed AGR, K^*_{pB}, PI_B, and *CO*'s public key explicitly binds the fingerprint to AGR, which specifies the purchased content. Thus, it is impossible for *CO* to frame the buyer. Also, the buyer uses a different K^*_{pB} and EA (PI_B) for the transaction with *CO* that prevents *CO* to frame the buyer by sending *BF* from a previous transaction. Therefore, framing an honest buyer by *CO* is not possible since he/she cannot forge any evidence.

C. Traitor tracing

Upon finding a pirated copy, *CO* executes the traitor-tracing algorithm of Nuida et al.'s codes [15] involving *MO* to trace the copyright violator(s). The traitor-tracing algorithm employs a scoring technique that outputs guilty user(s) with the highest score(s). The pseudo-identity of the guilty buyer is provided to the Arbitrator, who requests *CA* to reveal the real identity of the copyright violator.

D. Collusion Resistance

Nuida et al.'s codes [15] are c-secure with ϵ-error with $l \leq c$ (l is the number of colluders). In our system, we have considered $c = 3$. As long as l remains lower than c, the traitor-tracing algorithm is followed, the colluder can be identified successfully. Thus, the proposed scheme offers resistance against three colludes. The value of $c > 3$ can also be considered. However, this large value of c results in increased length m of the codeword, which will provide high collusion resistance but at a cost of lower content quality. The value of c can be chosen keeping in mind the desired security level of the system.

E. Privacy

The essential protection of the buyer's privacy is by taking advantage of the pseudo-identity generated with the help of *CA*. The buyer uses this pseudo-identity as his/her Ethereum address (EA) to purchase the content from *CO* via smart contract. To minimize the identifiability, a buyer is allowed to use different EAs (pseudo-identities) for different purchasing transactions. However, privacy provided to the buyer is revocable since pseudo-identity generated for a transaction with *CO* is registered with *CA*. Under *CA*'s existence, the buyer can keep his/her real identity unexposed unless he/she is found guilty of copyright violation by the Arbitrator.

F. Buyer Frameproofness

A malicious *CO* may attempt to collude with *MO* in a traitor-tracing protocol to frame an honest buyer for illegal redistribution. However, this attack can be disregarded since *MO* is an entity that is trusted by both *CO* and the buyer (as described in Section III-B2). In case *MO* acts maliciously and colludes with *CO* to frame the buyer, the accused buyer can request the Arbitrator to settle the dispute. The Arbitrator would retrieve the encrypted sk from the blockchain against TID provided by the buyer. The Arbitrator would

decrypt the encrypted sk using its secret key, and extract the fingerprint from secret embedding positions. Then, the Arbitrator would retrieve the hash of the fingerprint recorded against TID in the blockchain, and compare it with the hash of the extracted fingerprint. If both are identical, *CO* is proved guilty of framing an honest buyer for illegal redistribution. *CO* cannot deny of his/her malicious act since the hash of the fingerprint embedded in the buyer's content was recorded on the blockchain by *MO* during the fingerprint generation process (Section IV-D1).

G. Smart Contract Security

In this system, the messages exchanged between entities within smart contracts are time-stamped and encrypted using cryptographic keys. Only hashed data values and enciphered information are published on the blockchain. Therefore, an attacker cannot learn any significant information from the blockchain, thus, ensuring data integrity and confidentiality.

The off-chain communication between the IPFS network and the buyer is secured with a unique token that is communicated to him/her through the smart contract, and the existence of a TLS channel between two parties (as described in Section III-B2). Similarly, the off-chain communication between *CO* and *MO* is secure though public key encryption. Therefore, the communication is secure against man-in-the-middle attacks.

The perceptual hash of the content is stored in the blockchain that allows the buyer to compare the stored hash and the perceptual hash computed from the downloaded content to ensure that the purchased content is valid.

A buyer uses a pseudo-identity as his/her Ethereum address to purchase the content from *CO* via smart contract. This pseudo-identity is obtained from a cryptographic hash function. Thus, any attempt of an attacker to impersonate the buyer to obtain the fingerprinted purchased content is withstood by the collision resistance of the hash function. Furthermore, the impersonator cannot use the pseudo-identity of another buyer because he/she does not know the secret number r shared by the buyer with *CA*. Also, an encrypted unique token is provided to the buyer by *CO*. The attacker will not be able to obtain the content from the IPFS network because he/she does not have the private key of the buyer to decrypt the token.

All functions of the smart contract can only be accessed by the authorized entities (as described in Section III-B2). If the initiator of the function call is identified as an intruder, an error occurs and all states are reverted.

A possible attempt by an adversary to prevent publication of a valid transaction in the blockchain, e.g., by attempting a denial-of-service (DDoS) attack against a data usage event, is withstood by the proposed system as all transactions are recorded and stored on the public Ethereum blockchain in a decentralized manner. Thus, it is not subject to a single point failure, hacking or compromise. The Etheruem blockchain is robust to DDoS attacks as it is distributed globally and protected by thousands of mining nodes across the globe. Certainly, the attacker needs to have the control on more than a half of these mining nodes, which is assumed to be unfeasible.

VI. Conclusion

In this paper, a multimedia content distribution system is presented that combines blockchain technology, collusion-resistant fingerprinting and IPFS. All transactions and interactions between participating entities during the content purchase are controlled by Ethereum smart contract. The proposed system is a proof-of-concept, which can be extended in future by implementing and testing it on the Etheruem blockchain.

Acknowledgments

This work was partly funded by the Spanish Government through grants INCIBEC-2015-02491 "Ayudas para la excelencia de los equipos de investigaciòn avanzada en ciberseguridad", RTI2018-095094-B-C22 "CONSENT" and TIN2014-57364-C2-2-R "SMARTGLACIS."

References

[1] D. Megías and J. Domingo-Ferrer, "Privacy-aware peer-to-peer content distribution using automatically recombined fingerprints," *Multimedia Systems*, vol. 20, no. 2, pp. 105–125, 2014.

[2] A. Qureshi, D. Megías, and H. Rifà, "Framework for preserving security and privacy in P2P content distribution systems," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1391 – 1408, 2015.

[3] A. Qureshi, D. Megías, and H. Rifà-Pous, "PSUM: Peer-to-peer multimedia content distribution using collusion-resistant fingerprinting," *Journal of Network and Computer Applications*, vol. 66, pp. 180–197, 2016.

[4] D. Megías, "Improved privacy-preserving p2p multimedia distribution based on recombined fingerprints," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 2, pp. 179–189, 2014.

[5] D. Megías and A. Qureshi, "Collusion-resistant and privacy-preserving p2p multimedia distribution based on recombined fingerprinting," *Expert Systems with Applications*, vol. 71, pp. 147 – 172, 2017.

[6] M. Kuribayashi and N. Funabiki, "Decentralized tracing protocol for fingerprinting system," *APSIPA Transactions on Signal and Information Processing*, vol. 8, pp. 1–8, 2019.

[7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[8] Z. Meng, T. Morizumi, S. Miyata, and H. Kinoshita, "Design scheme of copyright management system based on digital watermarking and blockchain," in *COMPSAC'18*, vol. 02, 2018, pp. 359–364.

[9] C. Zhao, M. Liu, Y. Yang, F. Zhao, and S. Chen, "Toward a blockchain based image network copyright transaction protection approach," in *SICBS'19*, 2019, pp. 17–28.

[10] S. Zhao and D. O'Mahony, "Bmcprotector: A blockchain and smart contract based application for music copyright protection," in *ICBTA'18*, 2018, pp. 1–5.

[11] J. Kishigami, S. Fujimura, H. Watanabe, A. Nakadaira, and A. Akutsu, "The blockchain-based digital content distribution system," in *CCBD'15*, 2015, pp. 187–190.

[12] F. Miao, W. Yang, W. Fan, Y. Xie, Q. Guo, Y. You, Z. Liu, and L. Liu, "Digital copyright works management system based on dosa," in *CSAE'18*, 2018, pp. 179:1–179:9.

[13] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[14] "IPFS *is the Distributed Web*," https://ipfs.io/, accessed on August 24, 2019.

[15] K. Nuida, S. Fujitsu, M. Hagiwara, T. Kitagawa, H. Watanabe, K. Ogawa, and H. Imai, "An improvement of tardos's collusion-secure fingerprinting codes with very short lengths," in *AAECC'07*, 2007, pp. 80–89.

[16] J. P. Prins, Z. Erkin, and R. L. Lagendijk, "Anonymous fingerprinting with robust qim watermarking techniques," *EURASIP Journal on Information Security*, vol. 20, pp. 1–7, 2007.

[17] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Eurocrypt'99*, 1999, pp. 223–238.

[18] L. Weng, "Perceptual multimedia hashing," Ph.D. dissertation, Katholieke Universiteit Leuven, 2012, accessed on August 24, 2019.