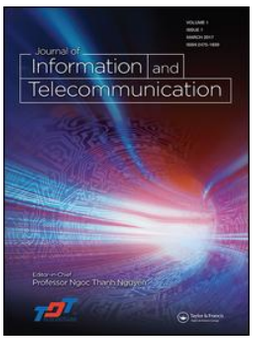


Taylor & Francis
Taylor & Francis Group

Journal of Information and Telecommunication

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tjit20

Traceability and ownership claim of data on big data marketplace using blockchain technology

Swagatika Sahoo & Raju Halder

To cite this article: Swagatika Sahoo & Raju Halder (2020): Traceability and ownership claim of data on big data marketplace using blockchain technology, Journal of Information and Telecommunication, DOI: 10.1080/24751839.2020.1819634

To link to this article: https://doi.org/10.1080/24751839.2020.1819634

Published online: 29 Sep 2020.

JOURNAL OF INFORMATION AND TELECOMMUNICATION
https://doi.org/10.1080/24751839.2020.1819634

Traceability and ownership claim of data on big data marketplace using blockchain technology

Swagatika Sahoo and Raju Halder

Department of Computer Science & Engineering, Indian Institute of Technology Patna, Patna, India

ABSTRACT

In the era of big data, modern data marketplaces have received much attention as they allow not only large enterprises but also individuals to trade their data. This new paradigm makes the data prone to various threats, including piracy, illegal reselling, tampering, illegal redistribution, ownership claiming, forgery, theft, misappropriation, etc. Although digital watermarking is a promising technique to address the above-mentioned challenges, the existing solutions in the literature are deemed to be incompetent in big data scenarios due to the following factors: V's of big data, involvement of multiple owners, incremental watermarking, large cover-size and limited watermark-capacity, non-interference, etc. In this paper, we propose a novel big data watermarking technique that leverages the power of blockchain technology and provides a transparent immutable audit trail for data movement in big data monetizing scenarios. In this context, we address all the crucial challenges mentioned above. We present a prototype implementation of the system as a proof of concept using Solidity on Ethereum platform, and we perform experimental evaluation to demonstrate its feasibility and effectiveness in terms of execution gas costs. To the best of our knowledge, this is the first proposal which deals with watermarking issues in the context of big data.

ARTICLE HISTORY
Received 15 June 2020
Accepted 2 September 2020

KEYWORDS
Big data; digital watermarking; blockchain; smart contract; access control

1. Introduction

Recent years have witnessed a dramatic increase in the generation of big data due to adoption of new technologies. Most of the enterprises now-a-days consider big data as a most significant resource and harness its power as a driving force to their business growth. This evergrowing demands of big data in rapidly changing competitive environment offers a new paradigm which encourages enterprises to adopt data monetization and initiates the establishment of large number of start-ups companies who sell and purchase our personal data on a daily basis. Few, among many others, include Datacamp, Datawallet, Dawex, etc. [1] There are many other situations where data monetization is an integral and indispensable part of a system in the form of data-as-a-service model (Terzo et al., 2013). Some interesting fields spawned and co-existing with the use of big data are machine learning, deep learning, artificial intelligence, data-science, etc., which may demand training dataset in a pay-per-use fashion. In this

CONTACT Swagatika Sahoo swagatika_1921cs03@iitp.ac.in

context, examples like census data collection shows its relevancy, where the task is outsourced to a large number of organizations in a hierarchical fashion to collect data locally at individual-levels and then to combine them together towards the higher levels, covering data-collection over a large geographical areas.

Although the above paradigm shift empowers individuals or organizations to sell their own data, this gives rise to major concerns related to our fundamental right to privacy and data security. In fact, data become prone to various threats, such as piracy, illegal reselling, tampering, illegal redistribution, ownership claiming, forgery, theft, misappropriation, etc. Surprisingly, a serious and subtle threat is that most of us are unaware about the existence of less-known data-brokers who gain profit by gathering, aggregating, analysing, hoarding, commodifying, trading or using personal data without our knowledge or consent (Parra-Arnau, 2018).

Digital watermarking has emerged as a promising technique to address these challenges (Halder et al., 2010). A watermark is considered to be some kind of information that is embedded into underlying data and is extracted later to prove the absence of above-mentioned activities. In general, the watermarking techniques consist of two phases: *Watermark Embedding* and *Watermark Verification*. During watermark embedding phase, a private key *K* (known only to the owner) is used to embed the watermark *W* into the original data. The watermarked data is then made publicly available. To verify the ownership of a suspicious data, the verification process is performed where the suspicious data is taken as input and by using the private key *K* (the same which is used during the embedding phase) the embedded watermark (if present) is extracted and compared with the original watermark information. Figure 1 depicts the basic watermarking technique.

There exist large number of watermarking approaches in the literature, targeting a variety of digital contents such as databases, images, audio, video, text data, etc. (Halder et al., 2010; Hartung & Girod, 1998; Kamaruddin et al., 2018). Broadly the watermarking

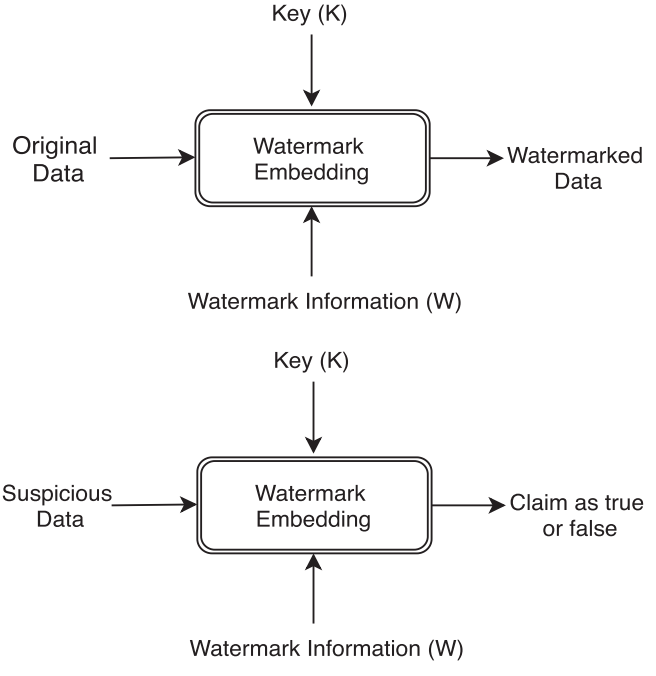


Figure 1. Basic watermarking technique.

on data can be categorized into distortion-based and distortion-free, depending upon whether the watermark is embedded into the data or is generated from the data. Existing proposals in both the categories are found unfit to adopt to the case of big data watermarking due to the following challenges: V's of big data, involvement of multiple owners, incremental watermarking, large cover-size and limited watermark-capacity, non-interference, etc (Chen et al., 2014). Notably, even though the four proposals (Iftikhar et al., 2017; Liu et al., 2017; Wangming, 2017; Yang et al., 2018) in the literature aim at watermarking specific kinds of big data (such as social network data, geographical big data, and large data algebraic graph) as cover, none of them addresses the above-mentioned challenges.

In recent years, blockchain technology (Nakamoto, 2009) has become a source of new hope with its broad spectrum of applications in practice. Examples include supply-chain, insurance, real-estate, financial services, and many more (Pilkington, 2016; Sahoo et al., 2019). We are witnessing its footstep in digital watermarking as well. Few recently proposed watermarking solutions using blockchain technology are reported in Bhowmik and Feng (2017), Billström and Huss (2017), Meng et al. (2018), and Zhao and O'Mahony (2018). Although, the use of blockchain in the above-mentioned proposals fulfils a common interest to improve the verifiability of the ownership in a distributed settings, unfortunately they are not meant for big data scenarios. In Yang et al. (2018), although the proposal keeps watermarked data sharing records in blockchain, this lacks in various aspects, such as this has not taken care of big data properties, access control, etc.

Motivated from the fact that the existing solutions in the literature are incompetent to be applied to the realm of big data watermarking, this paper proposes a novel approach that provides a transparent immutable audit trail for data movement in big data monetizing scenarios, by exploiting both watermarking and blockchain technologies power. To the best of our knowledge, this is the first proposal which deals with watermarking issues in the context of big data.

The major contributions in this paper are:

- We propose a novel watermarking technique for big data by leveraging the power of blockchain technology and smart contract, on top of existing watermarking technique at fine-grained level. The proposed system permits crucial operations in big data scenarios, including data collection, aggregation, storage, selling, and purchasing.
- We devise an access control mechanism to ensure the legitimate buyers, on receiving data-seller's permission, can get access to the target data-piece.
- We introduce a payment system, allowing data trading between sellers and buyers on big data marketplace. A review mechanism, with incentivization to buyers, is also put in place to attract more sellers and buyers in the system and to motivate them behaving honestly.
- We show in detail how our approach overcomes the present challenging factors in big data watermarking settings and allows a transparent immutable audit trail for data movement in big data marketplace.
- Finally, as a proof of concept, we present a prototype implementation of the proposed system using Java and Solidity on Ethereum platform, and report detailed experimental results.

This paper is a revised and expanded version of Sahoo et al. (2020). In particular, the improvements in this paper compared to Sahoo et al. (2020) are:

- An improved version of access control mechanism is presented, where we adopt proxy re-encryption technique to ensure security of the stored data.

- We consider a payment mechanism to allow trading of data among the stakeholders on big data marketplace.
- A review system is put in place, which allows buyers to evaluate data quality before buying it and to post reviews on the purchased data. In this setting, an incentive mechanism is also proposed to attract and motivate more participants to behave honestly in the system.
- We present a version control system that helps buyers to keep track of any changes in the older version of purchased data.
- We develop all newly added components along with an web interface. A rigorous experimental evaluation is also performed on all smart contracts, including tokenization process during data transfer.

The structure of the paper is organized as follows: Section 2 describes the related works in the literature. In Section 3, we identify the challenging factors involved in big data watermarking. Section 4 recalls some preliminaries on blockchain technology, interplanetary file system, and proxy re-encryption. The detail description of our proposed approach is presented in Section 5. The attack analysis is performed in Section 6. Section 7 presents proof of concept and experimental results. We discuss various aspect of our proposal w.r.t. the literature in section 8. Finally, Section 9 concludes our work.

2. Related works

Let us now discuss the state-of-the-art on big data watermarking and blockchain-based solutions in this research line. Over the past few decades, a large number of watermarking approaches on variety of digital contents are proposed. The digital assets referred by most of them include databases, images, audio, video, and text data (Halder et al., 2010; Hartung & Girod, 1998; Kamaruddin et al., 2018; Ng & Lau, 2005; Oh et al., 2001; Su et al., 2020). Let us restrict our discussion below only to big data context.

As we already mentioned before, only four approaches on big data watermarking are proposed in the literature in recent times (Iftikhar et al., 2017; Liu et al., 2017; Wangming, 2017; Yang et al., 2018). Although they aim at watermarking specific kinds of big data (such as social network data, geographical big data, and large data algebraic graph) as cover, none of them addresses the above-mentioned challenges. The authors in Iftikhar et al. (2017) proposed a reversible watermarking of social network data. The approa before watermark-encoding, adopted a data pre-processing phase where features of numeric and non-numeric datasets are selected and encoded into the generated watermark. Moreover, the Genetic Algorithm in case of the numeric dataset and two operations – hashing and permutations – for the non-numerical dataset are applied. Wangming (2017) presented the watermarking of large data algebraic graphs using a deep belief network. Aiming to solve the problem of interpretation attack, a zero-watermarking scheme for vector map, a type of geographical big data, is proposed in Liu et al. (2017) based on the feature points of vector maps.

Let us now briefly describe the recently proposed blockchain-based watermarking solutions to digital contents. Observe that none of them deal with big data (Bhowmik & Feng, 2017; Billström & Huss, 2017; Meng et al., 2018; Zhao & O'Mahony, 2018). In Zhao and O'Mahony (2018), the authors introduced BMCProtector, a prototype implementation based on blockchain and smart contracts, to protect music copyright issues and to ensure income rights/incentives to original holders or owners. The deployed smart contract is responsible to share the copyright

parameters of the music owners and to transfer cryptocurrency to their wallet when purchasing events take place. A new design approach for copyright management of image files based on digital watermarking and blockchain is proposed in Meng et al. (2018). The approach stores owner's digital signatures along with images' cryptographic hashes in the blockchain and the corresponding blocks information along with watermarked images in an interplanetary file system. In Billström and Huss (2017), the authors designed a proof of concept prototype which provides an online verification platform to verify the integrity of the recorded video. With the help of blockchain technology, the approach stores cryptographic hashes of video contents in a chronological chained link to establish an irrefutable database. A blockchain-based multimedia watermarking framework is proposed in Bhowmik and Feng (2017). This allows the retrieving of either the transaction trails or the modification histories of an image and preserves retrievable original media content identifying the edited/tampered regions. The framework proposed in Yang et al. (2018) facilitates the sharing of watermarked data, keeping records in the underlying blockchain, which is limited to numerical data only.

Since access control plays a crucial role in this context, let us now briefly highlight some related works on access control mechanism in blockchain. In Cruz et al. (2018), the authors introduced a role-based access control model using smart contract. In this model, the permission is assigned according to the characteristics and contextual information collected from the environment of the physical object. In Novo (2018), the author introduced a distributed access control model for blockchain in case of large-scale IoT systems consisting of billions devices over the globe. The access control policy is defined in the form of smart contract. The important components facilitating this access control in a distributed way are the introduction of access control manager nodes (light weight) and management hub nodes (high computation nodes) in the network. Observe that the model is suitable only for private blockchain network. In Zhang et al. (2019), the authors proposed a smart contract-based access control mechanism to achieve distributed and trustworthy access control for IoT systems. This implements both static access right validation based on predefined policies and dynamic access right validation by checking the behaviour of the subject. The approach facilitates dynamic validation by receiving misbehaviour reports, judging the misbehaviour and returning the corresponding penalty. There are few other proposals which address access control in blockchain network by incorporating access-control policies in the form of smart contracts (Alharby & van Moorsel, 2017; Zhang et al., 2019), etc.

In our system, we adopt data monetization to help trading or data marketing among data owners and data buyers. Let us discuss some relevant work done to maximize the revenue potential of user data. Banerjee and Ruj (2018) proposed blockchain-based solution to achieve fairness, efficiency, security, privacy in building a complete data marketplace. Blockchain is used here as a trusted third party applying some regulation through smart contract. In Kakushadze and Russo (2018), authors adopted blockchain technology to solve data provenance in the data market place like data malls and it introduce the keyless payment through blockchain. In Delgado-Segura et al. (2017), authors introduced a protocol to achieve fairness in the system through bitcoin transactions. The payment is done only after buyer receives the requested data. To enforce the fairness, bitcoin scripting language is used here. In addition to the above, few other solutions are also available to make trusted payment system (Choudhuri et al., 2017; Dziembowski et al., 2018; Ramachandran et al., 2018)

A comparative summary can be found in perspective of comparing existed papers with our proposed approach, is depicted in Table 1 where BC stands for Blockchain.

Table 1. A comparative summary w.r.t. literature.

	Metric							
Proposals	Is BC-based?	Cover type	Blockchain Type	Storage type	Support access control?	Dealing BigData?	Support payments?	Support Review/ Version Control?
Iftikhar et al. (2017)	N	Social network dataset	NA	NA	N	Y	N	N
Wangming (2017)	N	Algebric graph	NA	NA	N	Y	N	N
Liu et al. (2017)	N	Vector map	NA	NA	N	Y	N	N
Billström and Huss (2017)	Y	Video	Ethereum	Android devices + BC	N	N	N	N
Meng et al. (2018)	Y	Image	NA	IPFS + BC	N	N	N	N
Zhao and O'Mahony (2018)	Y	Music	Ethereum	IPFS + BC	Y	N	N	N
Bhowmik and Feng (2017)	Y	Multimedia	Ethereum	Media database server + BC	N	N	N	N
Yang et al. (2018)	Y	Numerical data	Fabric	BC	N	Y	N	N
Proposed Approach	Y	All types of data	Ethereum	IPFS + BC	Y	Y	Y	Y

3. Challenges in big data watermarking

Let us identify a number of challenging factors that lie with the big data watermarking (Chen et al., 2014; Halder et al., 2010):

(1) *Capacity*. The most important challenge in big data watermarking is *capacity*. It determines the optimum amount of data that can be embedded in a cover and the optimum way to embed and extract this information. Due to '*volume*' property of big data, this needs to be ensured that the embedded watermark is spread over all parts of the data. Moreover, if multiple marks are inserted into such large cover of big data, they should not interfere with each other.

(2) *Incremental watermarking*. The second property '*velocity*' of big data gives rise to another challenge: the generation of streaming data requires an adoption of incremental watermarking approach to ensure that the watermarking algorithm should consider only the newly added or modified data for the watermark, keeping the unaltered watermarked-data untouched.

(3) *Usability*. This is quite natural to assume that more than one actors participate in the process of big data collection. In addition to the original data-owners (who collects the data from the environment at individual levels), a number of data-collectors are also involved. The sole responsibility of data-collectors is to collect and aggregate data coming from either original data-owners or other data-collectors. Notably, in a more generic practical situation, data-collectors may also claim the ownership of the collected data, in addition to original data-owners. In such complex situation, the embedding of large number of ownership signatures, each corresponds to individual data-owner or -collector, may degrade the usability of big data.

(4) *Security*. The involvement of large volume of data and many ownerships make it a natural choice to embed multiple signatures into the big data. Such setting pushes watermarked-data towards a number of attacks, including subset attack, superset attack, collusion attack, etc.
(5) *Public verifiability*. The major drawback in private watermarking scheme is single-time verifiability, due to the possibility of revealing private parameters during the verification process to take place for the first time. In contrary, watermarking scheme based on public parameters overcome such limitations and anyone can verify the ownership at any time publicly. Big data watermarking requires a public treatment because of huge cover-size and multiple ownership involved, especially when big data can be split, aggregated and shared in pieces among many actors.
(6) *Trust*. Big data collection, storage, sharing, etc., requires the support of cloud-based infrastructure. In such case, trust is a concern when we are providing watermark solution to big data and at the same time relying on semi-trusted or even untrusted third party cloud service provider.
(7) *Traceability*. Traditional watermarking approaches do not support traceability. Although this may not be required in case of simple and restrictive scenarios involving single owner and limited sized data, this of course carries an important and impactful role in a complex scenarios like our case.

4. Preliminaries

Let us now briefly recall from (Benet, 2014; Egorov et al., 2017; Nakamoto, 2009) the preliminaries on blockchain technology, Interplanetary File System (IPFS) and Proxy Re-encryption, which we will refer often in the rest of the paper.

4.1. Blockchain and smart contract

With the advent of bitcoin (Nakamoto, 2009), its underlying blockchain technology has attracted huge attentions from both industry and academia in the recent years (Al-Jaroodi & Mohamed, 2019; Sahoo et al., 2019; Tsilidou & Foroglou, 2015). Blockchain is a type of distributed, transparent, trustless, publicly accessible ledger for maintaining a permanent and tamper-proof records of transactional data in a decentralized peer-to-peer network. Each of the nodes in the network maintains a copy of the ledger to prevent a single point failure and all copies are updated and validated simultaneously through a consensus mechanism. In particular, when a node performs any transaction, it is first broadcasted in the network. A set of nodes dedicate themselves in validating and collecting transactions in the form of blocks and compete in the network for mining them successfully into the existing blockchain by following any consensus algorithm, e.g. proof of work (POW), proof of stake (POS), etc (Narayanan et al., 2016). The addition of a new block into the existing blockchain is achieved by linking it using hash pointer generated from the content of the previous block. This ensures that the chain is never broken and that each block is permanently recorded.

A new addition to the power of blockchain technology comes with the support of smart contracts (Liu & Liu, 2019), an executable codes on blockchain written in high-level turing complete language (e.g. solidity). The role of smart contract is to remove all intermediary

untrusted third parties between the participating members and to automatically execute and enforce the terms of agreement between them.

4.2. Interplanetary file system

IPFS (Benet, 2014) is a new peer-to-peer hypermedia protocol for distributed file system which overcomes the single point failure and embraces the participation of untrusted parties in the system. IPFS has shown its advantage over existing cloud-based storage system in terms of decentralization and trust, by allowing data distribution across the globe and facilitating its sharing through content-addressed hyper-links. This storage system is immune to various attacks, such as altering, spoofing, man-in-middle, etc. (Zheng et al., December 2018). It is resistant to DDOS attacks due to content-addressing and content-signing. In recent times, IPFS has attracted a lot of use-cases, especially those where blockchain plays crucial roles (Produit, 2018). In particular, blockchain, along with IPFS, gives advantage of decentralized data storage while ensuring the credibility and immutability of stored data, thus eliminating the need of middle-man. In our solution, we propose to employ a private IPFS network, where data-owners can store data-units and fetch them whenever necessary. The encrypted URL links to the data stored on IPFS are stored on the blockchain smart contracts.

4.3. Proxy re-encryption

In this section, we discuss how our system can adopt a scenarios where, in addition to watermarked-data, few actors may also share, sell or purchase sensitive data. To this aim, we adopt proxy re-encryption technique (Egorov et al., 2017) that allows such file sharing by re-encrypting ciphertexts towards legitimate users via semi-trusted proxies, without them learning any information about the underlying message. Any proxy re-encryption scheme can be described by 5 algorithms (KeyGen, ReKeyGen, ReEnc, Enc, and Dec) as follows:

(1) ***KeyGen***(n): This function generates public–private key pair from n. For example, in case of identity based encryption, we pass the identity and system parameters as n.
(2) ***ReKeyGen***$(Pub_A, Priv_A, Pub_B, Priv*_B)$: This generates a re-encryption key $rencK_{A\rightarrow B}$ which is used by the proxy to convert a cipher text intended for A to another cipher text intended for B. The symbol $*$ denotes that private key of B may or may not be required depending on the algorithm.
(3) ***ReEnc***$(rencK_{A\rightarrow B}, C_A)$: This is executed by the proxy to convert the cipher text C_A (intended for A) into another cipher text C_B which can be decrypted by $Priv_B$.
(4) ***Enc***(m, Pub_A): This generates the cipher text C_A that can be decrypted by $Priv_A$.
(5) ***Dec***$(C_A, Priv_A)$: This decrypts the cipher text C_A to reveal the original data m.

Salient characteristics that distinguish different proxy re-encryption schemes are Directionality, Transitivity, Interactivity, and Collusion-Resistance.

5. Proposed approach

In this section, we present our proposed blockchain-based approach for big data watermarking. As data collection, storage and their maintenance in big data scenario involves

a number of participants, we categorize them into three kinds of actors based on their roles: Data-owner, Data-collector and Data-buyer. Data-owners are those persons or organizations who are responsible to generate data from the environment. The task of the data-collectors is to collect data either from data-owners or from other data-collectors and to transfer them further to another set of data-collectors. Since we are considering data as purchasing and selling items, there exist a number of data-buyers in the system who need data to buy either from data-owner or from data-collector. The overall system components are depicted in Figure 2 where the actors are interacting with a number of smart contracts, namely RegistrationSc, TransferOwnerShip, tokenSc, PaymentSc, AccessSc, ReviewSc and VcontrolSc, which provides various services in the system.

The proposed system consists of following six key phases:

Phase-1: Registration of Actors
Phase-2: Data Collection and Storage
Phase-3: Access Control
Phase-4: Payment Mechanism
Phase-5: Review and Rating System
Phase-6: Version Control System

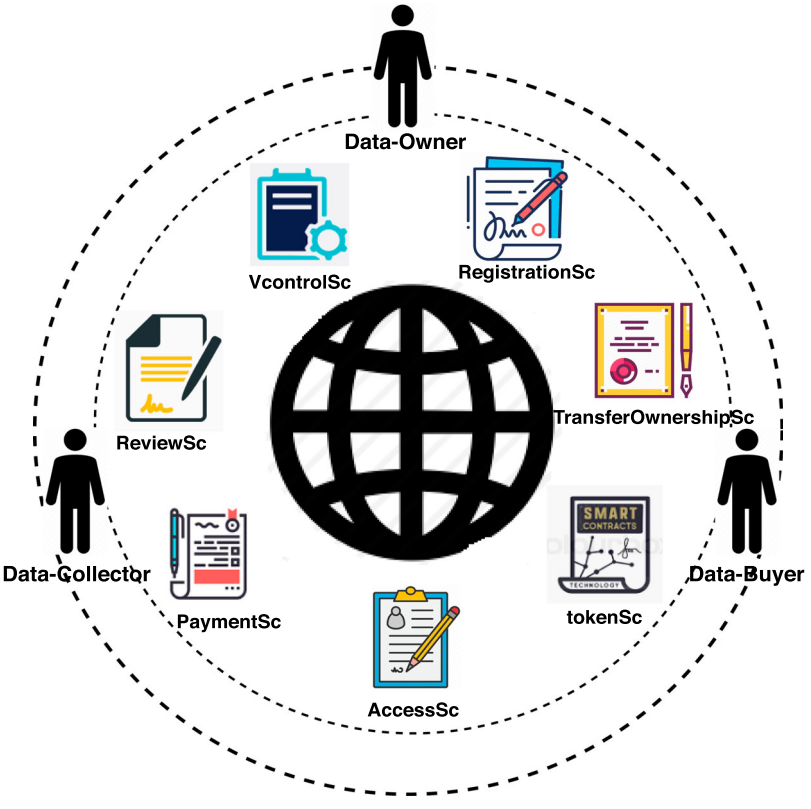


Figure 2. Overall system components.

5.1. Registration of actors

In order to participate in the system, all actors must pass through the registration phase in order to obtain login credential which needs to be used for authentication purpose later. The login credential includes unique identifier and password. Figure 3 depicts the interactions of data-owners o_l, data-collectors c_m, data-buyers b_i with RegistrationSc smart contract. RegistrationSc is responsible to store actors' details and to generate login credentials. This is to observe that the use of biometric-based authentication through national identity database improves the security of the registration process.

5.2. Data collection and storage

This is the core phase in our approach to ensure the copyright protection of big data. Unlike traditional centralized systems, our proposed system performs data collection completely in a decentralized manner. Multiple actors (owners and collectors) in the system are responsible to collect, aggregate, transfer and storage of data, which eventually form big data. In particular, data is collected from different sources by data owner and is being transferred to a number of collectors. The collectors, on the other hand, aggregate the incoming data and send it further to another set of collectors.

Let us now describe these activities involved in this phase separately:

5.2.1. Watermarking of owner's data

To avoid any malicious attack on data, an owner embeds her signature (as watermark information) into her own data. This allows the data-owner to claim the ownership of her data anytime in future. Let us describe a widely used database watermarking algorithm, also known as AHK algorithm (Agrawal et al., 2003; Agrawal & Kiernan, 2002), which could be suitably adopted in this case. Observe that, since the size of the data collected by individual owner is within a reasonable limit, we can apply any other suitable existing watermarking techniques (Halder et al., 2010) as well.

Given a database relation $R(P, A_0, A_1, \ldots, A_{\nu-1})$ where P is the primary key attribute and $A_0, A_1, \ldots, A_{\nu-1}$ are candidate attributes (numeric) used for marking. The embedding and

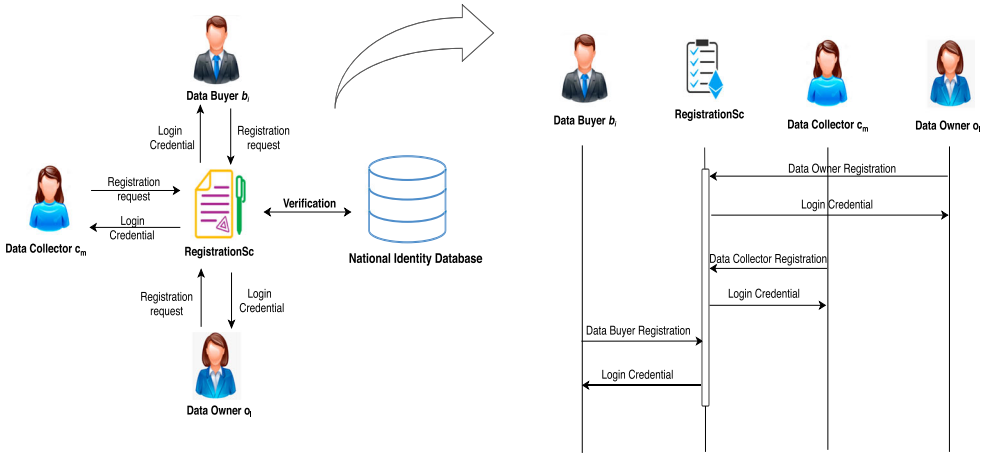


Figure 3. Actors' registration process.

detection of watermarks are depicted in Algorithms 1 and 2 respectively. In Algorithm 1, once a tuple *r* is found eligible for marking at step 2, the index *i* of the candidate attribute and its corresponding LSB position *j* are computed at steps 3 and 4, respectively. Step 5 invokes Mark function which flips the bit at *j*th LSB position in *i*th attribute depending on the hash of the private key concatenated with the primary key of the corresponding tuple. Observe that *H* and *F* are one-way hash and MAC functions, respectively.

The detection algorithm (Algorithm 2) considers a suspicious relation *S* and computes at step 6 the total number of tuples (denoted by *total-count*) already marked in the insertion algorithm. The 'Match' function at step 7 checks whether the marked bits at LSB position is present and accordingly computes the total number of successful detection (denoted by *match-count*). If *match-count* is more than the threshold τ (computed based on α and *total-count*), then the watermark detection is considered as successful (shown in steps 10–13).

Algorithm 1: Watermark Insertion Algorithm

```
   // The private key K and the parameters γ, ν and ξ are known only to the owner of the
      database.
   // 1/γ: Fraction of tuples marked.
   // ν: Number of attributes available for marking.
   // ξ: Number of least significant bits available for marking in an attribute.
 1 for tuple r ∈ R do
 2     if F(r · P) mod γ equals 0 then
 3         attribute_index i = F(r · P) mod ν;
 4         bit_index j = F(r, P) mod ξ;
 5         r · A_i = Mark(r · P, r · A_i, j);
 6     end
 7 end
 8 Mark(primary_key pk, number v, bit_index j) return number
 9     first_hash = H(K ∘ pk);
10     if first_hash is even then
11         set the j^th least significant bit of v to 0;
12     else
13         set the j^th least significant bit of v to 1;
14     end
15     return v;
16 return
```

Algorithm 2: Watermark Detection Algorithm

```
   // α (where 0 < α < 1): Test significance level.
   // τ: Minimum number of correctly marked tuples needed for detection.
 1 total-count = match-count = 0;
 2 for tuple s ∈ S do
 3     if F(s · P) mod γ equals 0 then
 4         attribute_index i = F(s · P) mod ν;
 5         bit_index j = F(s, P) mod ν;
 6         total-count = total-count + 1;
 7         match-count = match-count + Match(s ·P, s·A_i , j);
 8     end
 9 end
10 τ = Threshold(total-count, α);
11 if matchcount ≥ τ then
12     suspect piracy;
13 end
14 Match(primary_key pk, number v, bit_index j) return int
15     first_hash = H(K ∘ pk);
16     if first_hash is even then
17         Return 1 if the j^th least significant bit v is 0 else Return 0;
18     else
19         Return 1 if the j^th least significant bit v is 1 else Return 0;
20     end
21 return
```

5.2.2. Data storage

Data owners use IPFS to upload their watermarked-data in order to tackle various security issues, in contrast to the data storage in centralized architecture. IPFS returns a hash value corresponding to the uploaded data, which would be necessary in future to extract the data.

5.2.3. Registering watermarked-data

After successful storage of watermarked data, data-owners register them by invoking RegistrationSc with generated IPFS-hash and little more details about the data. The registration process of watermarked-data is same as that of the actor registration and an unique identifier for the registered-data is also generated in this case. Observe that IPFS-hash would be encrypted by owner's public key $P_k(Owner)$ before storing it into the smart contract's state variable. Moreover, in order to ensure the absence of any tamper during data-monetization, RegistrationSc also stores hash of the IPFS-hash during registration process. In the rest of the paper, the term 'data' always indicates 'watermarked-data'.

The above three activities are depicted pictorially in Figure 4.

5.2.4. Data transfer

This phase deals with the transfer of registered-data from owner to collector or collector to collector. TransferOwnershipSc is the smart contract which is responsible for this data transfer by changing the ownership information in the corresponding state variables of the smart contract. Observe that, in order to delegate the decryption right of the encrypted IPFS-hash from A to B during ownership transfer, it passes through proxy re-encryption technique which transforms the ciphertext (intended for A) into another (intended for B). This allows the data collector B to decrypt the IPFS-hash using her own private key and to access the data from the IPFS.

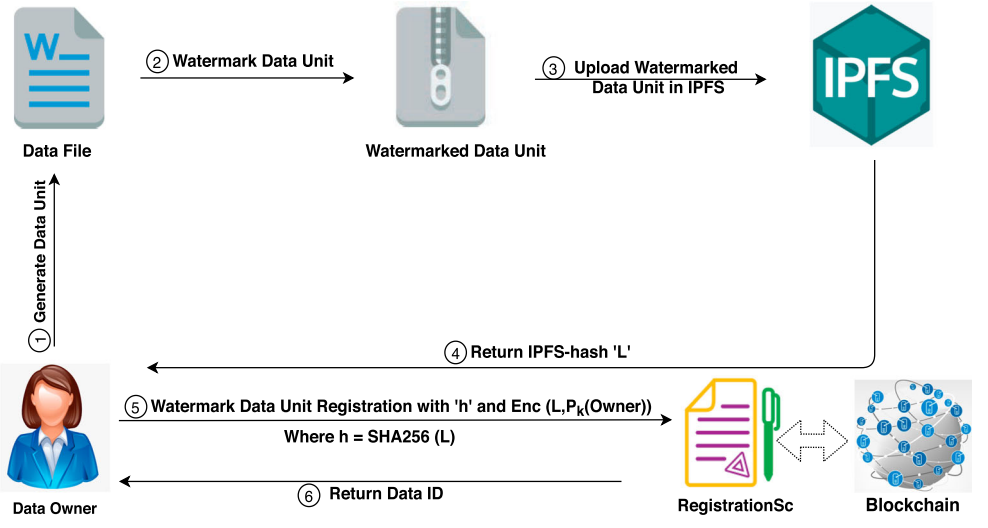


Figure 4. Data watermarking, storage, and registration.

As large number of data movements take place, in order to achieve scalability in the system, tokenization process is adopted at time of data transferring. This assigns unique token value against a group of data units and this token value is used to trace the movement of the group as a whole. In contrast, splitting mechanism is also introduced to allow transfer of only a part of it without disturbing their tracebility property. The smart contract tokenSc is responsible for performing these aggregate and splitting tasks. In the rest of the paper, we use the notations '$\langle\tilde{}\rangle$' and '$\succ\!\!\!\circ\{\tilde{}\}$' to denote aggregate and splitting operations, respectively.

Algorithm 3: TransferOwnership

Input : Sender s_i, Receiver r_j, Data-collection D
Output: Transfer ID id

Create a new token t to uniquely identify D;
for *all basic data unit IDs* $d \in D$ **do**
  if Owner(d) = s_i **then**
    Replace s_i by r_j;
    Append r_j to the list InheritOwners(d);
  else
    exit;
for *all token* $t_i \in D$ **do**
  if Owner(t_i) =s_i **then**
    Replace s_i by r_j;
    Append r_j to the list InheritOwners(t_i);
  else
    exit;
for *all splitted-token* $t_j \in t \succ\!\!\!\circ \phi$ **do**
  if Owner(t_j) =s_i **then**
    Replace s_i by r_j;
    Append r_j to the list InheritOwners(t_j);
  else
    exit;
Store the transfer ID id;
Pass through Proxy Re-encryption on the encrypted IPFS-hashes corresponding to D;
Return id;

Figure 5 exemplifies these operations and transfer operations. On the left side, the token '$tokenid_1$' refers to a group of three data units represented by the data identifiers

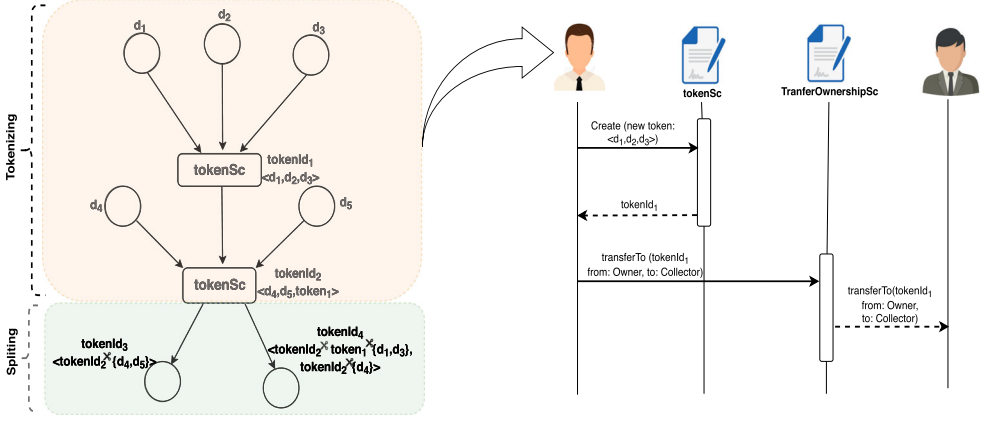


Figure 5. Tokenization, splitting and transfer.

'd_1', 'd_2' and 'd_3' respectively. Similarly, '$tokenid_2$' supports nested tokenization and consists of 'd_1', 'd_2' and '$tokenid_1$'. On the other hand, '$tokenid_4$' is obtained by performing splitting operation, which consists of 'd_4' and a part $\{d_1, d_2\}$ of '$tokenid_2$'. The creation of '$tokenid_1$' (shown within coloured tokenization box on the left side) and its transfer is shown on the right side in Figure 5.

The overall algorithm to perform two functionalities (data-transfer and data-traceability) of TransferOwnershipSc is depicted in Algorithm 3. The algorithm takes as inputs the unique identifiers of sender and receiver (which are generated in the registration phase) and the collection D of data for which ownership needs to be transferred. Observe that D may contain basic data units or previously generated tokens or splitted tokens representing other data-collections. The steps 2, 8 and 14 identify all basic data units, token units and spiltted token units, respectively, belonging to D. Steps 3, 9 and 15 verify whether the transfer request is issued by legitimate owner s_i by checking the current ownership of all data units, token units and spiltted token units, respectively, in D as per the record in the state variables. On successful verification, in steps 4–5, 10–11 and 16–17 replace and update the ownership. Observe that the function InheritOwners() helps to maintain a trace of ownership changes by appending all new owners to a list. Finally, the encrypted IPFS-hashes corresponding to D pass through proxy re-encryption in order to delegate the decryption right to the receiver r_j.

5.3. Complexity of InheritOwners algorithm

Let n_1, n_2 and n_3 be the number of basic data units, aggregated tokens and splitted tokens, respectively, in the data-collection D, where $|D| = n = n_1 + n_2 + n_3$. Therefore, steps 2, 8 and 14 iterate n_1, n_2 and n_3 times, respectively, resulting into the time complexity $O(n_1 + n_2 + n_3) = O(n)$. Steps 3, 9 and 15 check the list InheritOwners for the owner of the requested data units, token units, and splitted-token units respectively in order to verify whether the sender is legitimate owner or not. If verification is successful, the ownership is changed at steps 4, 10 and 16. Assuming m is the size of InheritOwners, this verification process exhibits $O(m)$ time complexity. So, the overall time complexity of the algorihm is $O(nm + nu) = O(n(m + u))$, where $O(u)$ is the time complexity of the proxy-reencryption which is carried over all n elements in D.

5.4. Access control

This section defines access control mechanism which enables data-owners or data-collectors to sell their data to registered data-buyers through a secure channel. Whenever a data-buyer wants to buy a chunk of data, the whole process must pass through an access control policy defined in AccessSc smart contract. Accordingly, data-owners have complete right to decide and give access-permission to the buyers. The access control mechanism is depicted in Algorithm 4 and it is shown pictorially in Figure 6. Let us discuss it in detail.

Algorithm 4: GetWatermarkedData

Inputs : Buyer's identity b_i, Data-collection D_j
Output: Encrypted IPFS-hash (intended for b_i) and hash of IPFS-hash corresponding to D_j

1 b_i requests to the smart contract `AccessSc` to purchase data D_j;
2 `AccessSc` interacts with the smart contract `RegistrationSc` to check whether b_i and D_j are registered.
3 **if** b_i *is valid buyer* **then**
4 **if** D_j *is registered data* **then**
5 *flag*:=true;
6 τ=getCurrentTime();
7 **Store** a new tuple $\langle D_j, b_i, \tau, flag \rangle$ into `LookTab`;
8 **Generate** an unique record ID r_k;
9 **else**
10 exit;
11 **else**
12 exit;
13 AccessSc invokes TransferOwnershipSc smart contract to retrieve owners-list of data D_j;
14 b_i selects the `Owner` o_l from owner-list and sends the request $\langle D_j, b_i, o_l \rangle$ to `AccessSc`;
15 `AccessSc` forwards buyer request $\langle D_j, b_i, o_l \rangle$ to the `Owner`;
16 `Owner` grants permission to `AccessSC`;
17 **if** $\exists r_k \in$ *LookTab* **then**
18 t_1=getCurrentTime();
19 **Return** Generates a unique code I;
20 Enters the seller information o_l, timestamp t_1, and unique code I into `LookTab`;
21 Sends I to the buyer and notifies about it to both TransferOwnershipSc and `Owner`;
22 **else**
23 exit;
24 b_i requests `AccessSc` for D supplying $\langle I, b_i, D_j, o_l \rangle$;
25 t_2=getCurrentTime();
26 **if** $t_2 - t_1 \leqslant \delta$ **then**
27 **if** *Verify*$(\langle I, b_i, D_j, o_l \rangle$, *LookTab*$)$==*success* **then**
28 **if** flag$\neq$ *false* **then**
29 Request for the IPFS information from TransferOwnershipSc corresponding to D_j owned by o_l and notify about it to `Owner`;
30 Receive the re-encrypted IPFS-hash (intended for b_i) and hash of the IPFS-hash corresponding to D_j;
31 Receive re-encrypt key from `Owner` o_l intended for b_i;
32 Set *flag*:= false;
33 **else**
34 exit;
35 **else**
36 exit;
37 **else**
38 exit;
39 Invoke PaymentSc;

When a buyer b_i requests for accessing the data collection D_j, AccessSc smart contract first checks whether both b_i and D_j are already registered or not through invocation of RegistrationSC smart contract as shown in the interactions ① and ②. After verification, AccessSc creates an entry in the look-up table LookTab with a new record ID r_k for the issued request. This is shown in step ③. The algorithmic steps corresponding to interactions ①–③ are specified in steps 1–12 in Algorithm 4.

In the interactions ④–⑥, AccessSc fetches from TransferOwnershipSc the list of sellers (either the original data-owner or the collectors) who have the copy of the requested data during its transfer from one to another. On receiving the list, buyer selects a seller o_l from whom she wishes to buy the data and sends her a permission request $\langle D_j, b_i, o_l \rangle$ through AccessSc. These are depicted in steps 13–15 in the algorithm.

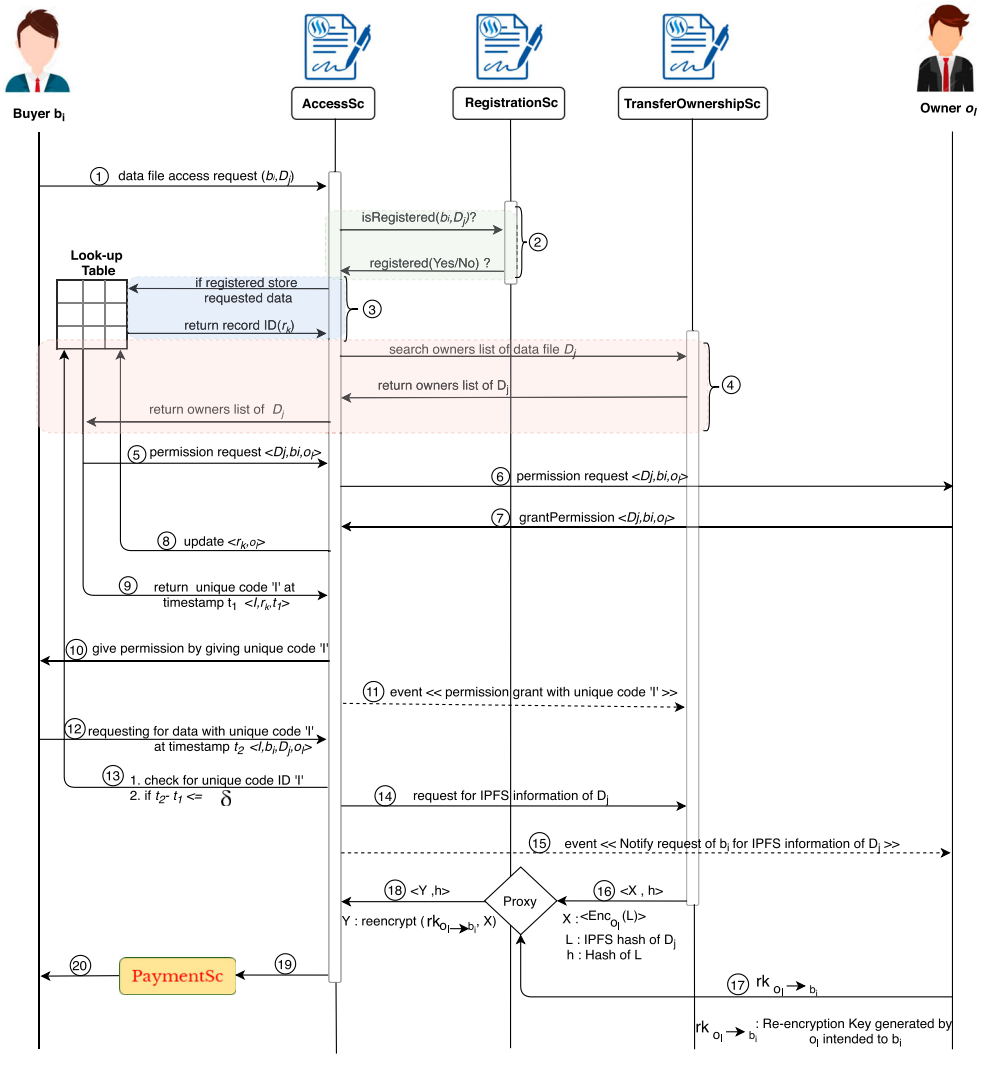


Figure 6. Access control mechanism.

To approve the access permission, the seller establishes a communication (interactions ⑦–⑩) to generate an unique code *I* at time instance t_1 against the buying request ID r_k, which will be notified to both the seller and buyer and will remain valid for δ time units. Observe that an event is also generated to notify TransferOwnershipSc about this code for the given seller-buyer pair. These are depicted in algorithmic steps 16–23.

On producing the same code *I* by the buyer at time t_2, AccessSc checks the time-gap $t_2 - t_1$ and compares the code with previously generated one from the LookTab table. On successful verification, AccessSc receives the IPFS information (i.e. encrypted IPFS-hash and hash of the IPFS-hash) intended for b_i using proxy re-encryption technique, and accordingly initiates the payment process by invoking PaymentSc. These are shown in the interactions ⑫–⑳ and in the corresponding algorithmic steps 24–39. The details of the payment process are discussed next.

5.5. Complexity of GetWatermarkedData algorithm

Assume that the data-collection D contains n number of elements and the algorithm uses hash-table (for example, use of mapping variable in solidity language) to store users and watermarked-data details during registration phase. Therefore, steps 3 and 4 exhibit the time complexity $O(n)$. Let m be the size of `LookTab` table. Each of steps 17 and 27 requires $O(m)$ time complexity to check the presence of a record in the `LookTab` table. Like step 4, steps 29 ans 30 also iterate n times over D resulting into a time-complexity of $O(n)$. Other steps in the algorithm have constant time complexity. Therefore, the overall time complexity of the algorithm is $O(n + m)$.

5.6. Payment mechanism

This section discusses trading between two parties – buyer and seller. This trading process is the sequence of activities during purchase or transfer of data-units between two-actors through AccessSc. The smart contract PaymentSc is deployed to ensure fairness, transparency, security, and privacy in the payment process by enforcing rules. The whole process is depicted in Figure 7.

Initially, AccessSc has two information Y and h. Y is the encrypted IPFS-hash which can be decrypted by only buyer's private key and h is the hash of the IPFS-hash. AccessSc first encrypts Y with a random key K, resulting into Z. In step 1, AccessSc sends Z to the buyer by encrypting it using buyer's public key Pk_B. On receiving Z, buyer deposits the payments to Paymentsc and, in addition, verifies the integrity of the received Z. These are shown in steps 2,3 and 4. On successful verification of its integrity, PaymentSc asks AccessSc for the random key K using which Y was encrypted and the hash of the original IPFS-hash and sends them to the buyer in steps 5–7. Thus, buyer can easily extract the original IPFS-hash L by following two decryptions in sequence, one by the random key K and another by her own private key Sk_B. Buyer can check the integrity of L by comparing its hash with the hash received from PaymentSc. In steps 8 and 9, buyer acknowledges to the PaymentSc to release the payment to seller's account.

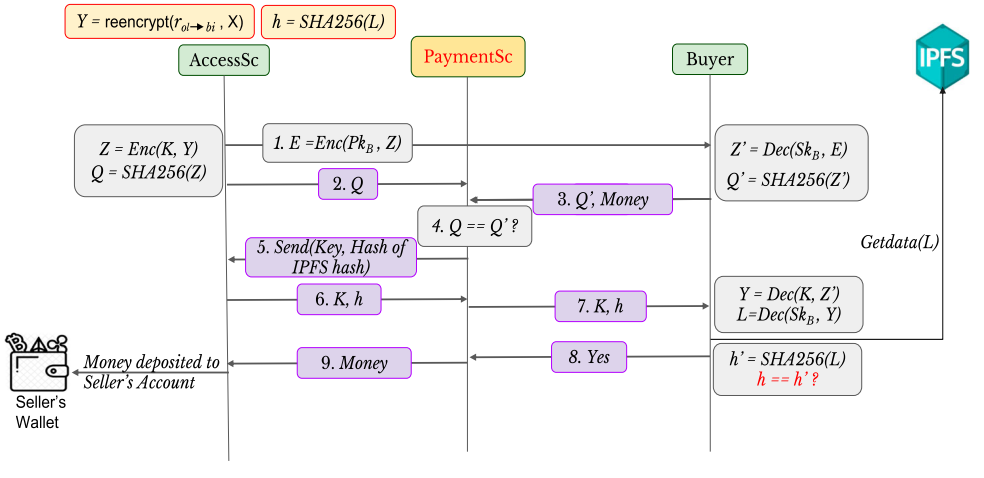


Figure 7. Payment scheme.

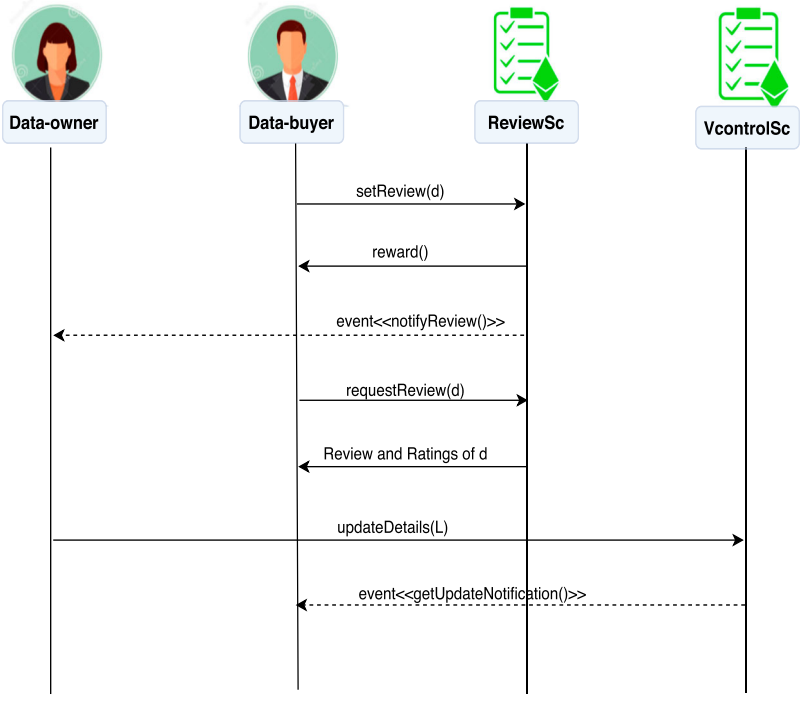


Figure 8. Review and version control system.

5.7. Review and rating system

This section focused on data-users reviews and ratings based on their experience in usage of data units. Before buying the data unit, it helps the data-buyers to view of its quality, reliability, resiliency, and integrity. In blockchain, reviews and ratings are stored with reviewers' identity by invoking ReviewSc smart contract which ensures that reviews will never be modified, manipulated or deleted by any entity. The overall review system is depicted in Figure 8. The legitimate data-buyer registers their reviews by invoking setReview(d) on a given data-unit d. These reviews are stored in the blockchain for further reference of new data-buyer, and it notifies to the data-seller of d. This would definitely help the data-seller to improve the data quality. When a new data-buyer wants to view the reviews and ratings of the data unit d, she contacts the ReviewSc smart contract invoking requestReview(d) function and receives the reviews and rating associated with d. This helps the data-buyer to make purchase decision for a given data unit. An incentive mechanism is introduced in the system to motivate the data-buyers for registering their reviews and ratings on data. The ReviewSc automatically calls the reward() function to incentivize the data-buyer. The reward amount should be declared by the data-sellers and deducted from seller's account.

5.8. Version control system

The primary goal of this phase is to track the document changes and deliver the updated version of the document to the data-buyer. For this purpose VcontrolSc, smart contract is

deployed in the system. Given a IPFS-hash L, for any kind changes on the data file by the data-owner through updateDeatils(L), the smart contract triggers a notification to the data-buyers through getUpdateNotification() shown in the interaction diagram 8. In such cases, data-buyers need not to pay any extra amount to get the updated document.

6. Attack analysis

The blockchain network encompasses nodes that send and receive transactions which are generated through smart contracts, and miners add approved transactions into the blockchain. Attackers generally look for a number of vulnerabilities that threaten the security of a system and may exploit them on the blockchain network. In this section, we discuss few relevant attacks and their preventive measures by the proposed approach.

- *Denial-of-Service (DoS) attack*. Denial-of-Service attack attempts to prevent legitimate users from availing services of a system by firing superfluous requests to targeted machines or resources keeping them throughout busy. DoS attacks may take place in our proposed system where attackers try to prevent data access services to other legitimate buyers. In particular, an attacker can initiate a DoS attack by following two means: (i) sending a large number of false requests intentionally to TransferOwnershipSc smart contract for data-access via AccessSc, or (ii) blocking payment channel by sending false requests to PaymentSc smart contract by untrusted stakeholder. Our proposed approach is made robust to this attack by introducing AccessSc smart contract which filters the incoming requests by verifying visitors' identities using the records stored in RegistractionSc smart contract. This allows to control the number the registered users in the system according to its capacity. Moreover, the system can also block users if they are found acting maliciously in the system. This is worthwhile to mention here that few blockchain platforms in the literature discourage DoS attackers by introducing payment mechanism in smart contract executions (e.g. Gas price in Ethereum Buterin, 2013).
- *Man in the middle attack*. Man in the middle attack is possible when an attacker intercepts the communication between two parties and tries to tamper the information which they are exchanging. To this aim, attackers usually steal unique ids, passwords, secret keys and other sensitive data which legitimate users possess. In our proposed system, attackers may attempt to obstruct the communication between buyers and owners and get access to the interaction ids which they present later to AccessSC in order to get data-access. This is impossible in the proposed framework because of the following factors: (1) The login mechanism using pre-approved credentials, (2) The sharing of interaction id through registered media, e.g. mobile, (3) One time use of generated interaction id for a given owner-buyer combination at a specific time-stamp, (4) The access restriction within a predefined time interval. In Algorithm 4, we achieve these by introducing a lookup table `LookTab` where all request information along with unique interaction id issued to a particular owner-buyer combination and the corresponding time-stamps are maintained. This prevents illegal requests from an attacker by verifying the entries in `LookTab`.
- *Attacks on Watermark:* Big data is highly susceptible to the following attacks: (a) Subset Attack, (b) Superset Attack, (c) Collusion Attack, (d) Value Modification Attack, etc. As

described in Section 5.2.1, our proposed system takes care of big data ownership by embedding watermarks in basic data blocks at fine-grained granularity level after collected by data-owners at individual levels. Therefore, existing watermarking techniques can easily be adopted to our case and the security of the embedded watermark relies completely on the security strength of the existing watermarking techniques. For example, the AHK algorithm (Agrawal et al., 2003; Agrawal & Kiernan, 2002) performs a probabilistic security analysis which shows its robustness against various forms of malicious attacks as well as benign updates to the data. In addition to this, we consider RegistractionSc to store hash values of basic data blocks (in the form of IPFS-links) during the registration phase to perform tamper detection. Since the blockchain network is pseudonymous, this is an advantageous step to weaken the collusion attacks.

7. Proof of concept and experimental results

We have implemented a prototype[2] to analyse the feasibility of the proposal. The programming languages we used in the implementation are Solidity and Python.

In the current implementation, we have provided complete functionalities in the form of Smart Contracts: RegistrationSc, TransferOwnershipSc, tokenSc, AccessSc, ReviewSc, VcontrolSc. Figure 9 illustrates the data structures used in different smart contracts. The smart contract compilation, deployment and executions are performed on a system configured with Intel processor, 1.80 GHz clock speed, 8 GB RAM and Windows 10 Professional 64-bit Operating System. In the experiment, we set the gas price to 2 *Gwei*, where 1 *Gwei* $= 10^9$ *wei* $= 10^{-9}$ *Ether* and 1 *Ether* $= 310$ USD.

In order to perform encryption of IPFS-hashes (during registration of watermarked-data) and their proxy re-encryption (during Data Transfer and Access Control), we used 'npre' [3] library which requires 'libssl-dev' and 'libgmp-dev' as its pre-requisites. This is a customized library written in Python and is a slightly refined version of the same algorithm in the charm crypto library. The interaction between proxy re-encryption off-chain computation and the smart contract is established using oraclize.[4]

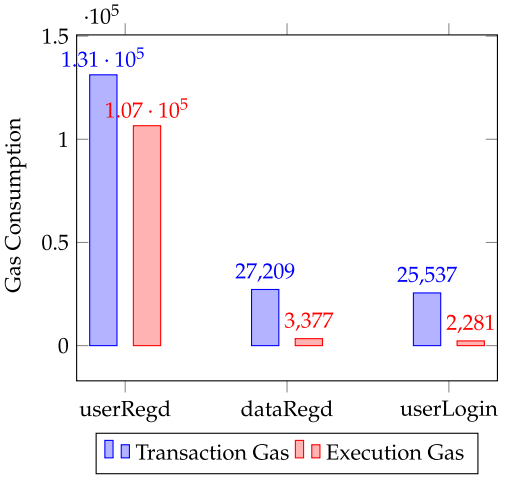
We conducted our experiment to record transaction gas, execution gas and actual cost for various functions involved in our system smart contracts. The gas costs are categorized into two: Deployment cost for a smart contract and Execution costs for various functions in the smart contract. Figure 10(a) shows the RegistrationSc smart contract functions and the costs involved in their execution. Observe that the function userRegd() is invoked when a new actor (user) wants to be part of the system and this requires $0.081 USD execution

```
struct data {
    string _data_name;
    uint _data_cost;
    string _data_specs;
    string _data_hash;
    uint _owner_id;
    uint _manufacture_date;
    address _data_owner_address;
    uint [] _owner_ids;
    uint [] trace_ids;
}

struct token {
    string username;
    uint userid;
    address owner;
    uint [] _data_ids;
    uint [] _token_ids;
}

struct track_data {
    uint _previous_owner_id;
    string _previous_owner_name;
    uint _data_id;
    uint _owner_id;
    string _owner_name;
    uint _timeStamp;
    string _owner_type;
    uint [] InheritOwners;
}
mapping(uint => track_data) public tracks;
```

Figure 9. A snippet of data structures used in RegistrationSc, TransferOwnershipSc, tokenSc.

Sl. No.	Functions	Task	Transaction Gas(*Wei*)	Execution Gas(*Wei*)	Actual Cost (*Ether*)	*USD* ($)
1	RegistrationSc	Deployment	3018659	2249599	0.006037318	1.87
2	userRegd	Execution	131227	106563	0.000262454	0.081
3	dataRegd	Execution	27209	3377	0.000054418	0.017
4	userLogin	Execution	25537	2281	0.000051074	0.015

(a)



(b)

Figure 10. Gas costs of different functions in RegistrationSc.

costs. Usually, data is made available for sale after its registration through dataRegd() and we have to pay $0.017 USD for the execution of this function. Similarly, when an actor login into the system, the userLogin() function needs to be performed, which takes $0.015 USD cost. The transaction and execution gas consumption against different functions are depicted graphical form on Figure 10(b).

Similarly, costs for the functionalities in TransferOwnershipSc, AccessSc, ReviewSc and VcontrolSc smart contracts are depicted in Figures 11(a), 12(a) and 13(a), respectively.

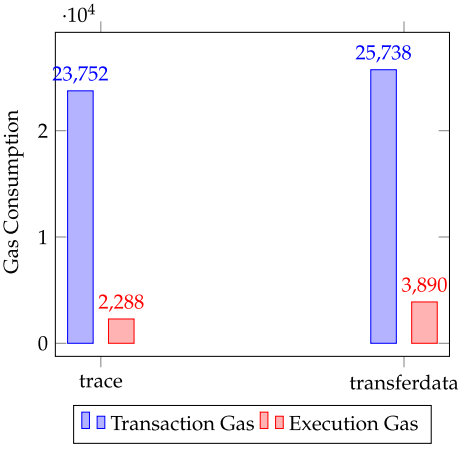
In Figure 14, we show the execution costs of TransferOwnershipSc with tokenization and without tokenization, using tokenSc smart contract. Observe that the gas cost varies linearly w.r.t. input data collection size.

8. Discussion

Although watermarking on big data is a serious concern nowadays, researchers have not paid much attention to it. As we already mentioned in Section 2, there are some existing solutions on big data watermarking (Iftikhar et al., 2017; Liu et al., 2017; Wangming, 2017; Yang et al., 2018) which address this problem in different directions, considering only specific types of big data (such as social network datasets, algebraic graphs, vector maps, numerical data). Still, they are not contemplating all problems and weaknesses involved in Big data Watermarking, which we have highlighted in Section 3. Let us now discuss the potential benefits of our system in overcoming these challenges:

Sl. No.	Functions	Task	Transaction Gas(*Wei*)	Execution Gas(*Wei*)	Actual Cost (*Ether*)	*USD* ($)
1	TransferOwnershipSc	Deployment	3754508	2806784	0.007509016	2.33
2	trace	Execution	23752	2288	0.000047504	0.014
3	transferdata	Execution	25738	3890	0.0000517	0.015

(a)

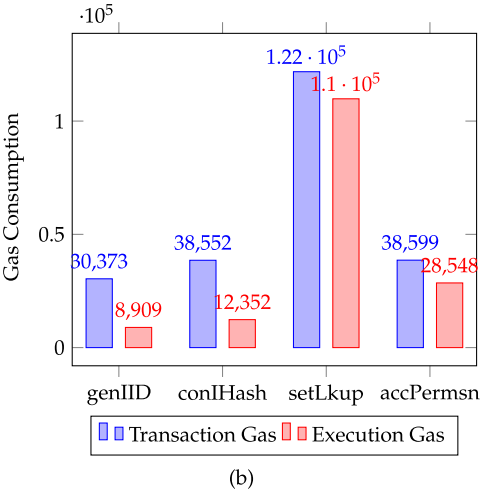


(b)

Figure 11. Gas costs of different functions in TransferOwnershipSc.

Sl. No.	Functions	Task	Transaction Gas (*Wei*)	Execution Gas (*Wei*)	Actual Cost (*Ether*)	*USD* ($)
1	AccessSc	Deployment	3508963	2624807	0.007017926	2.17
2	genIID	Execution	30373	8909	0.000060746	0.02
3	conlHash	Execution	38552	12352	0.000077104	0.02
4	setLkup	Execution	121793	109832	0.000243586	0.07
5	accPermsn	Execution	38599	28548	0.000077198	0.02

(a)

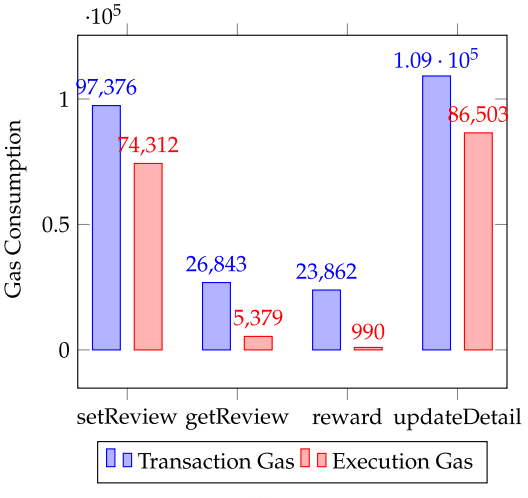


(b)

Figure 12. Gas costs of different functions in AccessSc.

Sl. No.	Functions	Task	Transaction Gas (*Wei*)	Execution Gas (*Wei*)	Actual Cost (*Ether*)	*USD* ($)
1	ReviewSc	Deployment	653438	451698	0.001306876	0.40
2	VcontrolSc	Deployment	441749	292537	0.000883498	0.27
3	setReview	Execution	97376	74312	0.000194752	0.06
4	getReview	Execution	26843	5379	0.000053686	0.02
5	reward	Execution	23862	990	0.000047724	0.01
6	updateDetail	Execution	109183	86503	0.000218366	0.07

(a)



(b)

Figure 13. Gas costs of different functions in ReviewSc and VcontrolSc.

(1) Authentication: To identify the legitimate user's participation in the system, we propose to use biometric-based authentication, which enhances the security of the system to avoid counterfeit. (2) Storage: We adopt a distributed file system (e.g. IPFS) to store the watermarked-data securely, to avoid the risk of a single point failure and to provide high integrity and global accessibility. This avoid the blockchain storage issue in terms of scalability. (3) Data security: We adopt a robust access control mechanism to enhance the system's security through AccessSc smart contract by restricting the access to unauthorized users. (4) Privacy: To maintain the privacy and confidentiality of data on the public blockchain, we use the proxy re-encryption technique to avoid the decryption during ownership transfer or data monetization. (5) Data provenance: We use TransferOwnershipSc smart contract to provide visibility of data movement, which gives the ability to trace the data by tracking the data ownerships and keeping the records of the ownership changes. This provides a transparent immutable audit trail for data movement in big data monetizing scenarios. (6) Scalability: As large number of data movement may occur at a time in the system, this gives rise to the scalability issue. Our proposed system achieve this by adopting tokenization or splitting mechanism using tokenSc smart contract. (7) Computation: To reduce the high extent redundant computations, we adopted the Off-chain computation concept to avoid high gas costs involved in On-chain computation. Off-chain computation can be combined with off-chain storage like as IPFS, as the

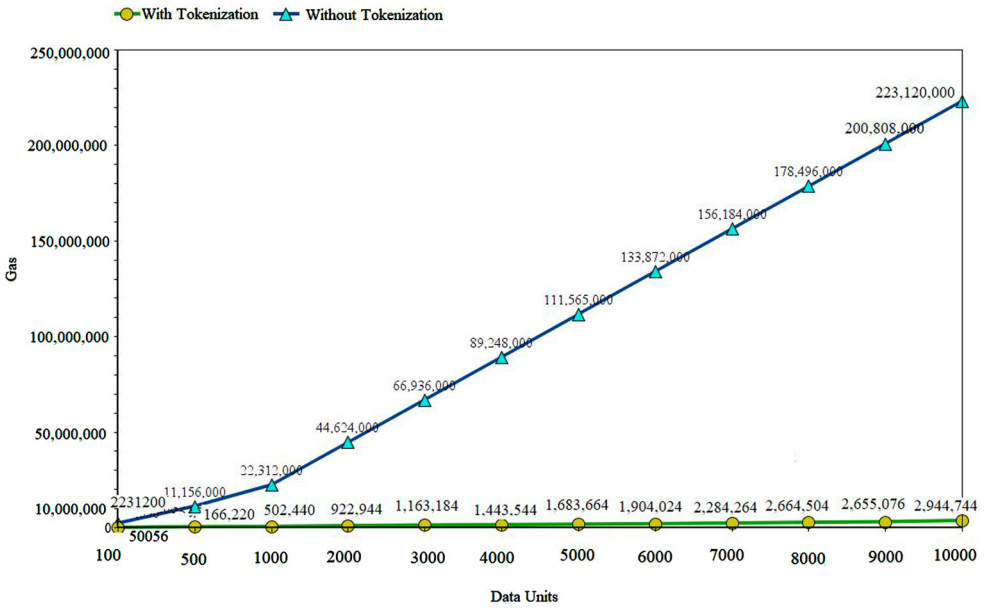


Figure 14. Gas costs for TransferOwnershipSc with tokenization and without tokenization w.r.t. input size.

means for reading inputs, recording outputs, etc. Moreover, we also incorporate off-chain re-encryption mechanism in order to save computational costs involved in on-chain settings. (8) Data monetization: We adopt a payment channel to make our system capable of dealing with fair trading among the stakeholders through PaymentSc smart contract. Apart from these potential benefits of our proposed system, we also identify scope for possible improvements to be considered as our future works. Let us highlight them below:

(1) Although our system proposes to use a biometric-based authenticaton system, in order to make the system more secure or to motivate anonymous participants to trust our system, we can adopt zero-knowledge proof (Feige et al., 1988) for credential verification. This technology works to preserve privacy in the system without revealing data and solves the problems which may occur due to eavesdropping. (2) Apart from data aggregation and splitting, as an alternative, we can also adopt a novel technique or some collaborative learning technique (Truex et al., 2018) that will ensure not to violate the property of data but contribute the model parameter instead of raw data by which blockchain nodes aggregate contributed models to build a global model. (3) The access control mechanism that we used in our system can be improvised by adapting some hierarchical dynamic-based access control mechanisms (Lan & Chunhua, 2013), which will give accessibility of data from level to level by applying a dynamic restriction policy. (4) In our system, we get reviews through ReviewSc smart contract, but we can apply some machine learning techniques to make it more useful by detecting fake reviews. Moreover, We can also include a penalty system for this. (5) Although this work does not consider any data quality, we can include this as an useful service in our system for which we may use suitable machine learning techniques to make the data more valuable before we watermark and store it in the IPFS.

9. Conclusion

In this paper, we propose a novel watermarking technique for big data by leveraging the power of blockchain technology and smart contract, on top of existing watermarking technique at fine-grained level. This allows the system to provide transparent immutable audit trail for data movement in big data marketplaces, addressing a number of common data breaches that are also highly probable in this paradigm. Interestingly, the approach is immune to several common attacks and its feasibility is established through a proof of concept. The experimental evaluation demonstrates its feasibility and effectiveness in terms of execution gas costs. To the best of our knowledge, this is the first proposal which deals with watermarking issues in the context of big data.

Notes

1. https://www.datacamp.com, https://www.datawallet.com/, https://www.dawex.com/en/
2. https://github.com/Swagatikasun12/BD-Monetization
3. https://github.com/nucypher/nucypher-pre-python/tree/master/npre
4. https://provable.xyz/

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Swagatika Sahoo is currently a Ph.D. student in the Department of Computer Science and Engineering, IIT Patna, India. She received her Bachelor of Technology and Master of Technology from Biju Patnaik University of Technology Rourkela, Odisha, India. Her research interests include blockchain technology, data privacy and security, machine learning.

Raju Halder received the doctoral degree from the Università Ca' Foscari Venezia, Italy, in 2012. Currently, he is an assistant professor in the Department of Computer Science and Engineering, IIT Patna, India. Before joining IIT Patna, he served as a post doctoral researcher at Macquarie University, Australia. He worked with the Robotics team at HASLab (University of Minho), Portugal, in 2016. Prior to his PhD, he had also worked as an associate system engineer at IBM India Pvt. Ltd. during 2007-2008. His areas of research interests include formal methods, blockchain technology, program analysis and verification, data privacy and security.

References

Agrawal, R., Haas, P., & Kiernan, J. (August 2003). Watermarking relational data: Framework, algorithms and analysis. *The VLDB Journal*, *12*(2), 157–169. https://doi.org/10.1007/s00778-003-0097-x

Agrawal, R., & Kiernan, J. (2002). Watermarking relational databases. In *VLDB '02: Proceedings of the 28th International Conference on Very Large Databases* (pp. 155–166).

Al-Jaroodi, J., & Mohamed, N. (2019). Blockchain in industries: A survey. *IEEE Access*, *7*, 36500–36515. https://doi.org/10.1109/ACCESS.2019.2903554

Alharby, M., & van Moorsel, A. (2017). A systematic mapping study on current research topics in smart contracts. *International Journal of Computer Science and Information Technology(ijcsit)*, *9*, 151–164. https://doi.org/10.5121/IJCSIT.2017.9511

Banerjee, P., & Ruj, S. (2018). Blockchain enabled data marketplace – design and challenges. *CoRR*, 2018.

Benet, J. (2014). IPFS - content addressed, versioned, P2P file system. *ArXiv*, *abs/1407.3561*.
Bhowmik, D., & Feng, T. (August 2017). The multimedia blockchain: A distributed and tamper-proof media transaction framework. In *Proceedings of 22nd International Conference on Digital Signal Processing (DSP)* (pp. 1–5). IEEE.
Billström, A., & Huss, F. (2017, August). *Video Integrity through Blockchain Technology*. KTH VETENSKAP OCH KONST.
Buterin, V. (2013). Ethereum white paper. *GitHub Repository*, *1*, 22–23.
Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, *19*(2), 171–209. https://doi.org/10.1007/s11036-013-0489-0
Choudhuri, A. R., Green, M., Jain, A., Kaptchuk, G., & Miers, I. (2017). Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery.
Cruz, J. P., Kaji, Y., & Yanai, N. (March 2018). Rbac-sc: Role-based access control using smart contract. *IEEE Access*, *6*, 12240–12251. https://doi.org/10.1109/ACCESS.2018.2812844
Delgado-Segura, S., Pérez-Solà, C., Navarro-Arribas, G., & Herrera-Joancomartí, J. (2017). A fair protocol for data trading based on bitcoin transactions. *IACR Cryptol. ePrint Arch.*.
Dziembowski, S., Eckey, L., & Faust, S. (2018). Fairswap: How to fairly exchange digital goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (pp. 967–984). Association for Computing Machinery.
Egorov, M., Wilkison, M., & Nuñez, D. (2017). Nucypher kms: decentralized key management system. *arXiv preprint arXiv:1707.06140* (pp. 1–5).
Feige, U., Fiat, A., & Shamir, A. (June 1988). Zero-knowledge proofs of identity. *Journal of Cryptology*, *1* (2), 77–94. https://doi.org/10.1007/BF02351717
Halder, R., Pal, S., & Cortesi, A. (2010). Watermarking techniques for relational databases: Survey, classification and comparison. *Journal of Universal Computer Science*, *16*, 3164–3190. https://doi.org/10.3217/jucs-016-21-3164
Hartung, F., & Girod, B. (1998). Watermarking of uncompressed and compressed video. *Signal Processing*, *66*(3), 283–301. https://doi.org/10.1016/S0165-1684(98)00011-5
Iftikhar, S., Kamran, M., Munir, E. U., & Khan, S. U. (2017). A reversible watermarking technique for social network data sets for enabling data trust in cyber, physical, and social computing. *IEEE Systems Journal*, *11*(1), 197–206. https://doi.org/10.1109/JSYST.2015.2416131
Kakushadze, Z., & Russo, R. P., Jr. (2018). Blockchain: Data malls, coin economies and keyless payments. *The Journal of Alternative Investments*, *21*(1), 8–16. https://doi.org/10.3905/jai.2018.21.1.008
Kamaruddin, N. S., Kamsin, A., Por, L. Y., & Rahman, H. (2018). A review of text watermarking: Theory, methods, and applications. *IEEE Access*, *6*, 8011–8028. https://doi.org/10.1109/ACCESS.2018.2796585
Lan, J., & Chunhua, G. (2013). A hierarchical access control technology for cloud storage. In *2013 International Conference on Computer Sciences and Applications* (pp. 35–40). IEEE.
Liu, J., & Liu, Z. (2019). A survey on security verification of blockchain smart contracts. *IEEE Access*, *7*, 77894–77904. https://doi.org/10.1109/Access.6287639
Liu, Y., Yang, F., Gao, K., Dong, W., & Song, J. (2017). A zero-watermarking scheme with embedding timestamp in vector maps for big data computing. *Cluster Computing*, *20*(4), 3667–3675. https://doi.org/10.1007/s10586-017-1251-3
Meng, Z., Morizumi, T., Miyata, S., & Kinoshita, H. (2018, July). Design scheme of copyright management system based on digital watermarking and blockchain. In *Proceedings of IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* (pp. 359–364). IEEE.
Nakamoto, S. (March 2009). *Bitcoin: A peer-to-peer electronic cash system*. http://bitcoin.org/bitcoin.pdf.
Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies*. Princeton University Press.
Ng, W., & Lau, H.-L. (2005). Effective approaches for watermarking xml data. In L. Zhou, B. C. Ooi, & X. Meng (Eds.), *Database systems for advanced applications* (pp. 68–80). Springer.
Novo, O. (March 2018). Blockchain meets iot: An architecture for scalable access management in iot. *IEEE Internet of Things Journal*, *5*(2), 1184–1195. https://doi.org/10.1109/JIOT.2018.2812239

Oh, H. O., Seok, J. W., Hong, J. W., & Youn, D. H. (2001). New echo embedding technique for robust and imperceptible audio watermarking. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)* (Vol. 3, pp. 1341–1344). IEEE.

Parra-Arnau, J. (2018). Optimized, direct sale of privacy in personal data marketplaces. *Information Sciences*, *424*, 354–384. https://doi.org/10.1016/j.ins.2017.10.009

Pilkington, M. (2016). Blockchain technology: Principles and applications. In F. Xavier Olleros & M. Zhegu (Eds.), *Handbook of research on digital transformations*. Edward Elgar.

Produit, B. (2018). Using blockchain technology in distributed storage systems.

Ramachandran, G. S., Radhakrishnan, R., & Krishnamachari, B. (2018). Towards a decentralized data marketplace for smart cities. In *2018 IEEE International Smart Cities Conference (ISC2)* (pp. 1–8). IEEE.

Sahoo, S., Fajge, A. M., Halder, R., & Cortesi, A. (June 2019). A hierarchical and abstraction-based blockchain model. *Applied Sciences*, *9*(11), 2343. https://doi.org/10.3390/app9112343

Sahoo, S., Roshan, R., Singh, V., & Halder, R. (2020). Bdmark: A blockchain-driven approach to big data watermarking. In P. Sitek, M. Pietranik, M. Krótkiewicz, & C. Srinilta (Eds.), *Intelligent information and database systems* (pp. 71–84). Springer.

Su, Q., Zhang, X., & Wang, G. (2020). An improved watermarking algorithm for color image using schur decomposition. *Soft Computing*, *24*(1), 445–460. https://doi.org/10.1007/s00500-019-03924-5

Terzo, O., Ruiu, P., Bucci, E., & Xhafa, F. (July 2013). Data as a service (daas) for sharing and processing of large data collections in the cloud. In *Proceedings of Seventh International Conference on Complex, Intelligent, and Software Intensive Systems* (pp. 475–480). IEEE.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., & Zhang, R. (2018). A hybrid approach to privacy-preserving federated learning. *CoRR*, abs/1812.03224.

Tsilidou, A., & Foroglou, G. (May 2015). Further applications of the blockchain. In *Proceedings of 12th Student Conference on Managerial Science and Technology*.

Wangming, Y. (November 2017). The digital watermarking algorithm based on the big data algebra graoh. In *Proceedings of International Conference on Robots & Intelligent System* (pp. 342–345). IEEE.

Yang, J., Wang, H., Wang, Z., Long, J., & Du, B. (2018, December 11–13). BDCP: A framework for big data copyright protection based on digital watermarking. *11th International Conference and Satellite Workshops, SpaCCS 2018*, Melbourne, NSW, Australia (pp. 351–360). https://doi.org/10.1007/978-3-030-05345-1_30

Zhang, Y., Kasahara, S., Shen, Y., Jiang, X., & Wan, J. (2019). Smart contract-based access control for the internet of things. *IEEE Internet of Things Journal*, *6*, 1594–1605. https://doi.org/10.1109/JIOT.2018.2847705

Zhao, S., & O'Mahony, D. (December 2018). Bmcprotector: A blockchain and smart contract based application for music copyright protection. In *Proceedings of International Conference on Blockchain Technology and Application* (pp. 1–5). ACM.

Zheng, Q., Li, Y., Chen, P., & Dong, X. (December 2018). An innovative ipfs-based storage model for blockchain. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (pp. 704–708). IEEE.