PLOS ONE

RESEARCH ARTICLE

Secure and reliable blockchain-based eBook transaction system for self-published eBook trading

Jeonghee Chi, Jangyeon Lee, Nakyung Kim, Jeewoo Choi, Soyoung Park*

Department of Software, Konkuk University, Seoul, South Korea

* soyoungpark@konkuk.ac.kr

Abstract

As eBook readers have expanded on the market, various online eBook markets have arisen as well. Currently, the online eBook market consists of at least publishers and online platform providers and authors, and these actors inevitably incur intermediate costs between them. In this paper, we introduce a blockchain-based eBook market system that enables self-published eBook trading and direct payments from readers to authors without any trusted party; because authors publish themselves and readers purchase directly from authors, neither actor incurs any intermediate costs. However, because of this trustless environment, the validity, ownership and intellectual property of digital contents cannot be verified and protected, and the safety of purchase transactions cannot be ensured. To address these shortcomings, we propose a secure and reliable eBook transaction system that satisfies the following security requirements: (1) verification of the ownership of each eBook, (2) confidentiality of eBook contents, (3) authorization of a right to read a book, (4) authentication of a legitimate purchaser, (5) verification of the validity and integrity of eBook contents, (6) safety of direct purchase transactions, and (7) preventing eBook piracy and illegal distribution. We provide practical cryptographic protocols for the proposed system and analyze the security and simulated performance of the proposed schemes.

Citation: Chi J, Lee J, Kim N, Choi J, Park S (2020) Secure and reliable blockchain-based eBook transaction system for self-published eBook trading. PLoS ONE 15(2): e0228418. https://doi.org/10.1371/journal.pone.0228418

Editor: He Debiao, Wuhan University, CHINA

Received: September 26, 2019

Accepted: January 14, 2020

Published: February 3, 2020

Data Availability Statement: All relevant data are within the manuscript.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Introduction

An eBook, also known as an electronic or digital book, is a digitally released version of a book, often consisting of text and images and available on electronic devices. According to statista.com [1], e-books have become popular among American book readers– 20 percent of book readers in the United States stated that they read more eBooks than hard copy books, and 23 percent read about the same number of hard copy books and eBooks. In South Korea, many eBook specialized platforms including ridibooks [2], Millie [3], and Series [4] have recently launched. Demand for eBooks has been increased steadily, and the number of self-published eBooks has risen dramatically in the last decade. In 2017, there were a total of over one million print and eBooks self-published in the United States, of which 879,600 were print books and 129,600 were eBooks [5]. Currently, the eBook market system consists in essence of publishers,

online platform providers (or shops), and authors; the online eBook platform serves as a trusted party for secure distribution and management of digital contents. The platform contracts with the author or publisher and hosts the sales of the digital contents; then the sales revenue is distributed between the two. Therefore, an intermediate fee occurs for the publisher and distributor.

With the advent of YouTube, publishing, sharing and subscribing to personal creations has become very popular today. Even for eBook contents, a platform that anyone can freely publish, share and sell personal eBook contents is necessary. Many web-novels in Korea, mostly serialized fiction, are published regularly two or three times a week. The first few series can be previewed for free, and after the story raises its awareness, the next series will have to be paid for subscription. An open eBook trading platform will allow more people to freely share and sell their personal novels. The goal of our research is to develop a real-time eBook trading platform over a peer-to-peer (P2P) network where anyone can publish their own eBook contents and buy a right to read the contents from authors. Because the authors publish themselves and receive direct payments from readers, authors incur no royalty costs for publishers, authors only pay a minimum fee for using the eBook platform, and readers benefit from lower book prices. Here, the most difficult issue is that the proposed model allows the sale of personal eBook contents rather than just sharing them, so the proposed model should be able to ensure the reliability of eBook contents, copyright protection and the safety of eBook purchase transactions. With emerging of the blockchain technology [6–15], the blockchain-based smart contract [16], data sharing [17], and DRM technology [18] have been proposed. The blockchain, a distributed database that maintains a continuously increasing list of ordered records, is trusted for correctness and availability because all data recorded in the blockchain is publicly validated.

In this paper, we introduce a secure and reliable blockchain-based eBook market system that enables self-published eBook trading and direct payments from readers to authors without outside trusted parties; users can be either authors or readers. However, owing to the trustless environment, no one can take responsibility not only for the validity, ownership, and intellectual property of digital contents but also for the safety of purchase transactions. Our proposed eBook market system satisfies the following security requirements: (1) verification of the ownership of each eBook, (2) confidentiality of eBook contents, (3) authorization of a right to read a book, (4) authentication of a legitimate purchaser, (5) verification of the validity and integrity of eBook contents, (6) safety of direct purchase transactions, and (7) prevention of eBook piracy and illegal distribution.

The key strategies to achieve our goal are that (1) every eBook information containing the author information is digitally signed with the author's private key, deployed over the network, and recorded on the blockchain named *B_Chain*; (2) every eBook content is encrypted with its own book key; (3) an eBook's content is divided into many small book pieces and stored in a book repository as encrypted; (4) only a legitimate purchaser can obtain the book key and the set of encrypted pieces of the purchased book; (5) a buyer can restore a part of the content of a randomly selected book piece during the purchasing process and verify the validity and integrity of the content; (6) every eBook purchase transaction is signed by both the buyer and the author, deployed over the network, and recorded on the blockchain named *C_Chain*, and (7) every purchased eBook's content is re-encrypted and stored with a new book key so that only the buyer's service application can restore the content.

Since every eBook information is recorded on *B_Chain* and shared among users in a distributed way, the ownership and copyright of the book can be publicly proved. Buyers and authors directly conduct the eBook purchase transactions, during which the buyer can validate the book's content and the book key for access; the buyer requests a randomly selected piece of

book and decrypts it with the received book key to confirm the content. In addition, the system compares the hash value of the restored book piece with the registered hash value, thereby validating both the book key and the eBook content during the purchasing process. The proposed challenge-and-response protocol combines both advantages of hash-based message authentication and provable data possession (PDP) that generates proofs of possession for randomly selected sample data blocks. The system reveals only a randomly selected piece of book not the entire eBook content. With this revealing process, comparing the hash values is enough for the content verification, a complex homomorphic verifiable cryptographic technique used in the PDP is unnecessary. In addition, the buyer can verify if the revealed book texts make sense or not. It is very effective and suitable for real-time content verification.

When the buyer and the author complete the purchase transaction by mutual agreement, *C_Chain* for the transaction is updated, and deployed over the network so that no purchase transactions can be either forged or repudiated. The buyer with a valid purchase transaction recorded on *C_Chain* can download all the book pieces of the purchased eBook from the book repository. That is, if a buyer acquires only a book key during the book purchase process but does not complete the purchase process, a valid blockchain cannot be generated, and it is impossible for the buyer to obtain the eBook content from the book repository; that is, the book key alone is useless, which in turn protects the eBook content from a malicious buyer.

Traditional copyright protection technologies focus on detecting the data owner or tracking illegal distributers using with watermarking and fingerprinting techniques. In contrast, the proposed model protects the illegal distribution of copyrighted contents by preventing even legitimate buyers from accessing to the original content. All downloaded eBook contents are newly encrypted by the buyer's service application and stored in the buyer's local storage. In other words, buyers of the same eBook content receive differently encrypted copies of the book, and the book key they obtained during the eBook purchase is subsequently useless. Because the original plain content of the purchased eBook is neither exposed nor stored anywhere, it is impossible for even the legal purchaser to redistribute or resale the original content to others. Even if an encrypted version stored in the buyer's local storage is distributed to others, the other service applications cannot reveal the encrypted content because the service applications cannot produce a valid book key for the content.

Consequently, this paper provides a new real-time eBook trading platform that supports self-publishing and direct payments between users on a P2P network without trusted parties. The proposed model uses blockchain technology to effectively protect copyrights on paid contents, and securely manage direct payment transactions.

- All eBook information is publicly verified with *B_Chain*.
- All eBook purchase transactions are publicly verified with *C_Chain*.
- Only validated eBooks registered in *B_Chain* are uploaded to the book repository. Only legitimate buyers can download eBooks from the book repository with their purchase transactions recorded in *C_Chain*. Thus, only verifiable eBooks are shared among users on the network and only legitimate buyers can obtain encrypted pieces of books for their purchased books.

The proposed model suggests a practical purchasing protocol with content verification suitable for real-time services.

- The purchase process includes not only making a payment for the purchased book, but also securing the right to read the book. Therefore, validating the book key provided by the author and verifying the encrypted content stored in the book repository is essential to the

buyer. The proposed hash-based challenge-and-response protocol effectively resolves this issue without restoring the entire encrypted content before the purchase is completed.

- The purchasing process is interactive between the buyer and the author and requires signatures from both parties. Hence, a valid purchase transaction can only be created by mutual agreement between the author and buyer.
- The proposed model effectively prevents piracy and illegal distribution on paid content.
- All purchased eBook contents are stored re-encrypted in the buyer's local device. Access to the original content of the purchased eBook is unavailable, which prevents from being copied or distributed.

The rest of this paper proceeds as follows. In Section 2 and Section 3, we review related works and briefly describe the elliptic curve cryptography [19–21] closely associated with the security of our system. In Section 4, we provide concrete cryptographic protocols that satisfy all the requirements mentioned above. We analyze the security and efficiency of the proposed protocols in Section 5, briefly describing our implemented eBook transaction system including its average execution time for main operations such as eBook registration, purchase, eBook download and storage; we analyze the system according to the size of eBook content and the level of difficulty used in the blockchain generation. Finally, we conclude our paper in Section 6.

Related work

Peer-to-peer (P2P) systems allow all peers to distribute and share data directly with each other by willingly contributing their resources. Since year 2003, P2P file sharing traffic has surpassed web traffic and has been continuously grown up. Many P2P systems including Napster, Gnutella, BitTorrent, and Skype have been developed and still widely used [22]. Because various types of copyrighted digital content are easily distributed and shared among peers on the network, many copyright protection technologies have been proposed to prevent the illegal distribution of copyrighted content to unauthorized users. Most copyright protection technologies exploit digital watermarking [23–25] and fingerprinting technologies [26–28]. Digital watermarking embeds a watermark into the content that can be used to check the source of the content. The watermarking is classified into copyright watermarking and fingerprinting watermarking. The copyright watermarking inserts the copyright holder's identification into the content in order to declare the copyright holder. But it cannot be used to trace an illegal distributer. On the other hand, the fingerprinting watermark insert a unique user identification into the content so that it can be used to track an illegal distributer [29]. A buyer-seller watermarking schemes [30–32] that incorporates both watermarking and fingerprinting mechanisms to protect the rights of both the buyer and the seller have been proposed. All of the above techniques are useful for detecting content owners or tracking illegal distributors, but do not prevent illegal distribution itself because the content copy holders can possess the same original contents in their local devices.

When accessing data (or content) stored on a remote peer or server, the integrity of the data should be verified first. That is, it is required to make sure that the remote server possesses the original data. There are three ways to prove the integrity of data stored in a remote server. The first is to check the hash value of data using the message authentication code (MAC) algorithm [33] or the hash function [34]. A verifier (or client) calculates the hash value of the original data and compares it with the hash value given from the remote server or with a previously calculated hash value. The main drawback of this approach is that it needs to access the entire

data to calculate the hash value. The second approach is to use a third-party auditor (TPA) [35], which carries out all auditing process. On behalf of the data owner, the TPA issues an audit message (or challenge) to the remote server to validate the integrity of the data stored in the remote server. Any modification to the data has been occurred, the data owner is notified about those changes and then validates his or her data. The disadvantages of this method are the need for the third party as well as the communication channel with the third party. The last is the provable data possession (PDP) [36] model that generates probabilistic proofs of possession by sampling random sets of blocks from the server without retrieving the original data. Data is divided into small blocks and stored in remote server. Using with a homomorphic verifiable tag, a data owner generates a random challenge for specific data blocks, then the server responds with the proofs of possession of the blocks. The owner can validate the integrity of his or her data by verifying the validity of the proof without retrieving the original data or accessing to the entire data.

With the emergence of blockchain technology [6–15], publica [37]—a peer-to-peer publishing platform based on blockchain technology—was recently introduced; it enables authors to publish their works and allows direct transactions between readers and authors. A reader purchases a book token, which is an access key to a particular book, with cryptocurrency, and all purchase contracts are managed by blockchain. All works are stored in decentralized book storage, and readers with valid book tokens can download the books to their devices. However, the system does not allow for buyers to verify the validity of a purchased work or book token during the purchase. As a trusted party, Publica (or a publisher associated with the book) is responsible for the validity of the book content and the book token. However, in self-publishing in a trustless environment, readers must validate the content they have purchased and the corresponding access keys during the purchase. In addition, it is necessary to strictly prevent illegal copying and redistribution of downloaded eBook contents.

Elliptic curve cryptography

We first describe the public-key cryptographic schemes, based on elliptic curve cryptography, that provide basic security in our system.

(1) Elliptic curve over F_p

Let F_p be a prime finite field so that p is an odd prime number, and let $a, b \in F_p$ satisfy the following Eq (1).

$$4 \cdot a^3 + 27 \cdot b^2 \not\equiv 0 \pmod{p} \tag{1}$$

An elliptic curve $E(F_p)$ over F_p, which we use for our basic public-key cryptosystem (encryption and digital signature), consists of the set of points $P = (x, y)$ for $x, y \in F_p$ to the Eq (2).

$$E : y^2 \equiv x^3 + a \cdot x + b \pmod{p} \tag{2}$$

together with an extra point O called the point at infinity. For a given point $P = (x, y)$, x is called the x-coordinate of P, and y is called the y-coordinate of P. The number of points on $E(F_p)$ is denoted by $\#E(F_p)$, an additive Abelian group. Suppose that $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are points in $E(F_p)$: The following addition rules are defined:

- $O + O = O$
- $P + O = O + P = P$
- $P + (-P) = O$, where $-(x, y) = (x, -y)$

- If $x_1 \neq x_2$, $P + Q = (x_3, y_3)$ where

$$x_3 \equiv \lambda^2 - x_1 - x_2 \,(\mathrm{mod}p), y_3 \equiv \lambda^2 \cdot (x_1 - x_3) - y_1 \,(\mathrm{mod}p), \text{ and}$$

$$\lambda \equiv \begin{cases} \dfrac{y_2 - y_1}{x_2 - x\_1} \;\; (\mathrm{mod}p) \;\; P \neq Q \\ \dfrac{3 \cdot x_1^2 + a}{2 \cdot y_1} \;\; (\mathrm{mod}p) \;\; P = Q \end{cases}$$

The domain parameters of $E(F_p)$ are denoted as $T = (p, a, b, G, n, h)$ consisting of an integer p specifying the finite field F_p, two elements $a, b \in F_p$ specifying the above elliptic curve, a base point $G = (x_G, y_G)$ on $E(F_p)$, a prime n that is the order of G, and an integer h, which is the cofactor $h = \#E(F_p)/n$.

(2) Elliptic curve key pair

For a given elliptic curve, an elliptic curve private-and-public key pair $(K^-, K^+) = (d, Q)$ is generated as follows:

1. randomly select an integer d in the interval $[1, n\text{-}1]$
2. calculate $Q = dG$
3. output (d, Q)

(3) Elliptic curve integrated encryption scheme (ECIES)

For a given domain parameter T, let two users be U and V, and let the private-and-public key pairs of U and V be (K_u^-, K_u^+) and (K_v^-, K_v^+), respectively. Suppose that U sends a message M encrypted with V's public key. The encryption algorithm is performed as follows:

1. select a random secret integer k and generated a point $R = kG = (x_R, y_R)$
2. compute $Q = k{\cdot}K_v^+ \; (= k{\cdot}d_V{\cdot}G) = (x_Q, y_Q)$ and $z = x_Q$
3. derive a symmetric encryption key and a MAC key $k_E \mid k_M = KDF(z)$, where $KDF(\cdot)$ is the key derivation function
4. compute $EM = E(k_E, M)$ and $D = MAC(k_M, EM)$, where $E(\cdot)$ is a symmetric encryption algorithm and $MAC()$ is a MAC scheme
5. output a ciphertext $C = R \mid EM \mid D$

The decryption algorithm of V for a ciphertext C is performed as follows:

1. compute $Q = K_v^-{\cdot}R \; (= d_V{\cdot}k{\cdot}G = k{\cdot}d_V{\cdot}G = k{\cdot}K_v^+) = (x_Q, y_Q)$ and $z = x_Q$
2. derive a symmetric encryption key and a MAC key $k_E \mid k_M = KDF(z)$
3. compute $MAC(k_M, EM)$, and if $MAC(k_M, EM) \neq D$, output 'failed'
4. decrypt $M = D(k_E, EM)$ where $D(\cdot)$ is a symmetric decryption algorithm

(4) Elliptic curve digital signature algorithm (ECDSA)s

For the given domain parameter T and the private-and-public key pairs of two users U, V same as above, the signing operation of U for a message M, denoted as $Sig(K_u^-, M)$, is performed as follows:

1. select a random secret integer k and generated a point $R = kG = (x_R, y_R)$
2. $r = x_R \bmod n$, if $r = 0$, return to step 1)

3. e = an integer conversion of $H(M)$, where $H(\cdot)$ is a cryptographic hash function. If $\lceil \log_2 n \rceil < 8$ (hash length), then e = an integer conversion of the leftmost $\lceil \log_2 n \rceil$ bits of $H(M)$
4. $s \equiv k^{-1} \cdot (e + r \cdot K_u^-) \pmod{n}$
5. output (r, s)

The verifying operation V for a signature $S = (r, s)$ is performed as follows:

1. e = an integer conversion of $H(M)$, if $\lceil \log_2 n \rceil < 8$ (hash length), then e = an integer conversion of the leftmost $\lceil \log_2 n \rceil$ bits of $H(M)$
2. $u_1 \equiv e \cdot s^{-1} \pmod{n}$ and $u_2 \equiv r \cdot s^{-1} \pmod{n}$
3. compute $R = (x_R, y_R) = u_1 G + u_2 K_u^+$. If $R = O$, then output 'invalid'
4. $v \equiv x_R \pmod{n}$. If $v = r$, output 'valid', otherwise, output 'invalid'

Blockchain-based eBook transaction system

System configuration, assumptions and security requirements

The proposed eBook transaction system consists of user, service application, eBook content, digital coin, blockchain, book repository, and P2P network. Users can be either authors or readers, and each user can use all the services of the proposed system only through the service application; the service application, denoted as *SA*, is dedicated software that conducts all functions specified in our system. We give a detailed description of the *SA* in the next subsection. An eBook is defined as a digital text document that can be sold independently, and it can be a stand-alone book or part of a series; the digital coin in the proposed system is a cryptocurrency used to pay for eBook purchases. Existing cryptocurrencies such as Bitcoin [6–7] and Ethereum [8] can be used, but our implemented service application used our developed cryptocurrency. All eBook information and eBook purchase transactions are distributed over the network and maintained on blockchains named *B_Chain* and *C_Chain*, respectively. A book repository, which acts as a content server, stores all published eBook contents in an encrypted manner. Every eBook is divided into small pieces of book content, and each piece of book is encrypted with its own book key. The repository stores only the encrypted pieces of eBooks but does not store any private or secure information including book keys; the repository also has an authentication logic layer that verifies whether an upload or download request is valid or not.

Fig 1 shows the configuration of the proposed system and an illustrated scenario of eBook registration and purchase. Any user can either publish a new eBook using a *SA* or purchase and read an eBook using the application. Once an author *A* completes a new book, *A* registers the book. The book information is broadcast to all users on the network, and *B_Chain* is updated. Encrypted pieces of the book are uploaded to the book repository; the current book list of every *SA* is also then updated. If a reader *B* wants to buy *A*'s published book, *A* and *B* complete that purchase protocol between them; *B* can validate the book before completing the purchase with a book key obtained during the purchase process. When *A* and *B* complete a valid transaction, the transaction deploys on the network, and *C_Chain* is updated. *B* can request the encrypted pieces of the book from the book repository, which then validates the purchase transaction and *C_Chain*. If all are valid, *B* downloads the book pieces, re-encrypts them with a new key, and stores them locally.

For practical use of the proposed system, the following assumptions are made:

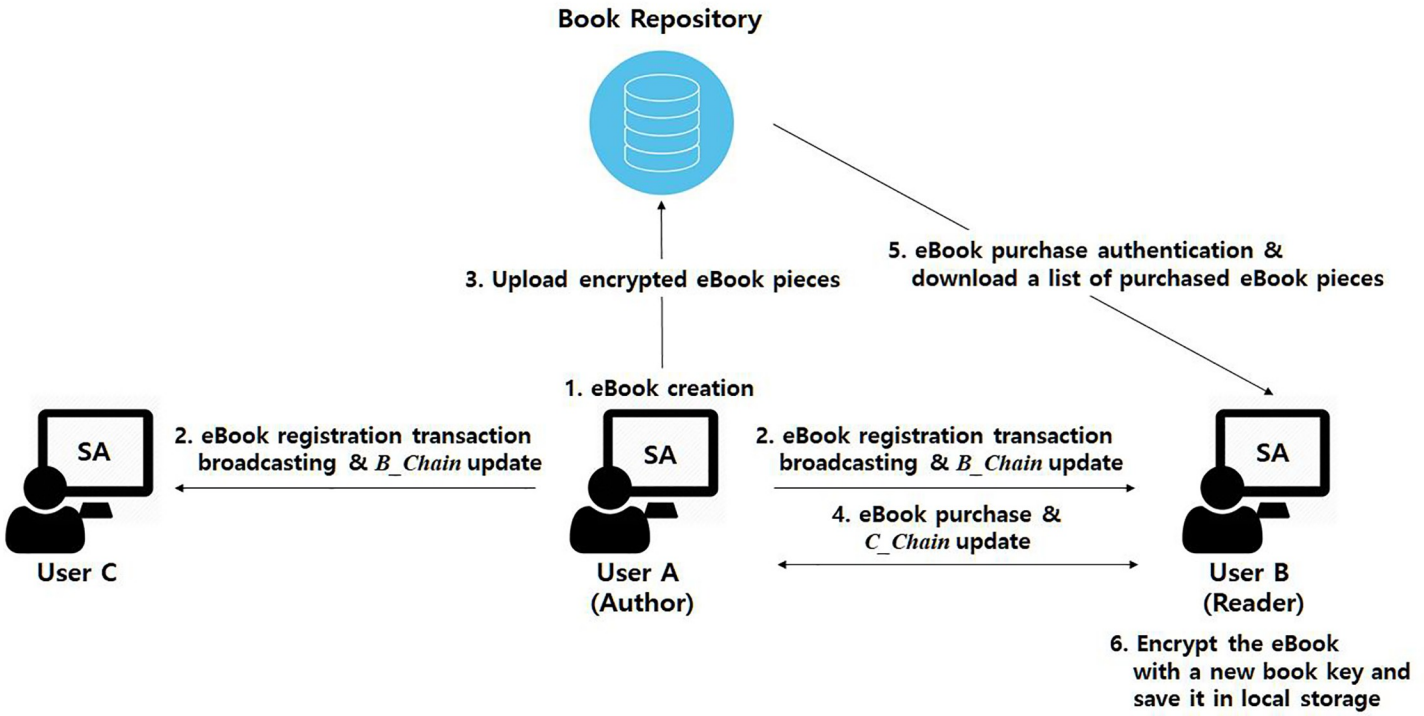

Fig 1. eBook transaction system configuration and the operation of eBook publishing and purchasing.

https://doi.org/10.1371/journal.pone.0228418.g001

- The *SA* is trusted. The provider of the proposed system is responsible for keeping the system available and safe, and for resolving any claims incurred while using the proposed system. But, the provider is not responsible for the published content itself.
- The *SA* has a master secret key programmed in a secure manner, and it is impossible for users to know the master key.
- Only eBook purchase is allowed. eBook rental, which grants a right to read for a particular period, is disallowed.
- Only authors have the ownership of their created eBook contents; eBook readers have only a "right to read" for their purchased contents. Thus, readers cannot resell their purchased books.
- The proof of working (or mining block) for a blockchain update can be accomplished by any user on the network, but for efficient and practical use of the scheme, the author is basically responsible for the proof of work.

SA is the most crucial element in our system; Fig 2 shows the *SA* configuration. *SA* consists of four main modules: communication, security, eBook transaction, and user interface. The communication module is responsible for all kinds of P2P communications between users on the network including the book repository. The security module is responsible for performing fundamental cryptographic protocols including public key encryption and decryption, digital signature and verification, and symmetric key encryption and decryption. SHA256 [38–39] is used for a hash algorithm, and ECDSA is used for a basic digital signature algorithm. ECIES is used for a public key encryption and decryption algorithm, and AES [40–41] with the key size of 256 bits is used for a symmetric key encryption and decryption algorithm. The eBook transaction module, which plays a key role in the proposed system, performs three main actions:

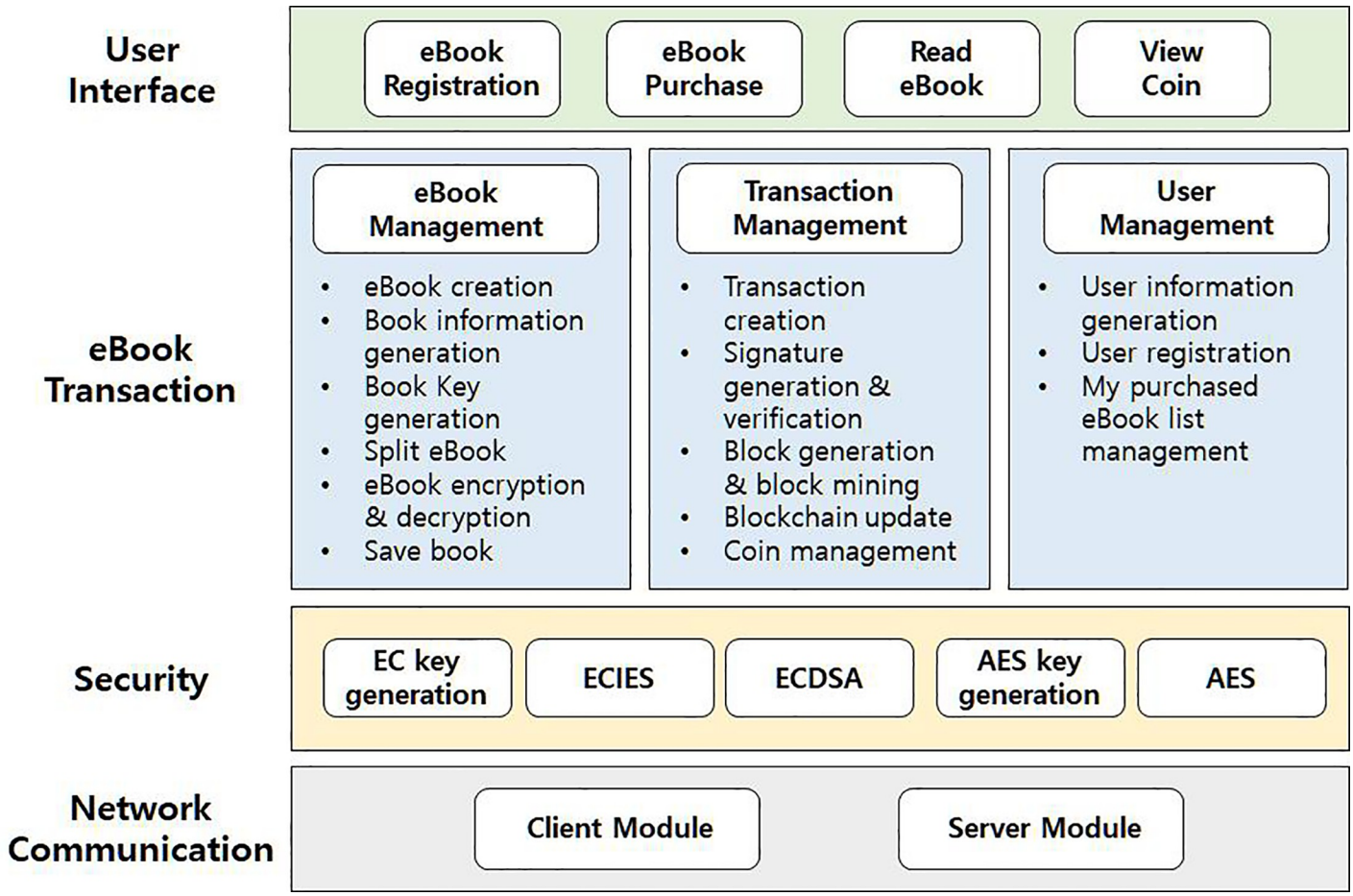

Fig 2. The configuration of a service application.

https://doi.org/10.1371/journal.pone.0228418.g002

eBook management, transaction management, and user management. The eBook management relates to when a new eBook is created, generates its book key, divides the book into the pieces and encrypts them, and so on. The transaction management creates blocks for eBook registration transactions and eBook purchase transactions, updates the corresponding blockchain, and manages users' digital coin. The user management module registers users and managers their book lists. Finally, the user interface module provides all kinds of user interfaces for eBook registration, purchase, reading, and showing current coin.

Next, we describe the format of transactions for eBook registration and eBook purchase, and explain the blockchain update process. The format of both types of transactions are given in Table 1.

Table 1. Format of transactions for eBook information registration and eBook purchase.

(a) The format of eBook registration transaction		(b) The format of eBook purchase transaction	
Field	**Description**	**Field**	**Description**
ID	The hashed result of this transaction	**ID**	The hashed result of this transaction
Signature of author	The author's digital signature on the transaction content	**Signature of buyer**	The buyer's digital signature on the contract information
Content	The eBook information	**Signature of author**	The author's digital signature on the contract information
		Contract	The detailed eBook purchasing information

https://doi.org/10.1371/journal.pone.0228418.t001

Table 2. Format of block.

Field	Description
Block hash	The hash value of this block
ID	The sequence number of this block
Previous block hash	The hash value of the previous block in the chain
Merkle tree root	The root hash value of the Merkle hash tree
Timestamp	The creation time of the block in the chain
Difficulty	The level of difficulty for the proof-of-work
Nonce	A nonce for the proof-of-work
Transactions	All transactions in the block

https://doi.org/10.1371/journal.pone.0228418.t002

A new block containing eBook registration transactions collected during a predefined time interval is periodically created, and the block is added to the current *B_Chain*. In contrast, in order to enable real-time eBook purchases, a new block containing the purchase transaction is created whenever a new purchase transaction occurs, and the block is added to the current *C_Chain*. Originally, the hash value of the new block is determined as the root hash value of the Merkle tree composed of the hash values of transactions included in the block. For *C_Chain*, because a block usually contains a single transaction, the transaction hash value becomes the block's root hash value.

As assumed above, every author is supposed to generate the proof-of-work about a current block. The proof-of-work for each block is to find a nonce that can generate a hash value starting with 0 as many as the number specified in a predefined difficulty field. The format of each block is given in Table 2.

The algorithm for updating blockchain *add*(*chain*, *transaction*) and the algorithm for verifying transaction *verify*(*chain*, *transaction*, *block_id*) are given below:

add(*chain*, *transaction*) performs

1. TID_i = a set of hash values of transactions collected for the *i*-th time interval;
2. compute *tid* = *H*(*transaction*);
3. if (verify the *signature* of *transaction*) then $TID_i = TID_i \cup \{tid\}$;
4. if (block creation time) then
   A. p*revious_block_hash* = *block_hash* of *chain*;
   B. compute *root_hash* = *merkle_tree_root_hash*(TID_i);
   C. find *nonce* such that *H*(*previous_block_hash* | *nonce* | *root_hash*) satisfies the difficulty;
   D. generate a new *block* and update *chain* = *H*(*previous_block_hash* | *nonce* | *root_hash*);

verify(*chain*, *transaction*, *block_id*) performs

1. find $block_{id}$ with *block_id* in *chain*;
2. compute *tid* = *H*(*transaction*) and find a transaction T_{id} with *tid* in $block_{id}$;
3. obtain a public key *pk* from *contract* of T_{id};
4. if (verify the *signature* of *transaction* with *pk*)
   A. generate *new_TID* = $TID-\{tid_{id}\} \cup \{tid\}$ where tid_{id} is the hash of T_{id};

B. compute $new\_root\_hash = merkle\_tree\_root\_hash(new\_TID)$;

C. $previous\_block\_hash_{id} = previous\_block\_hash$ of $block_{id}$;

D. $nonce_{id} = nonce$ of $block_{id}$;

E. compute $new\_block\_hash = H(previous\_block\_hash_{id} \mid nonce_{id} \mid new\_root\_hash)$;

F. if ($new\_block\_hash = block\_hash$ of $block_{id}$) output *true*;

else output *false*;

The main goal of the proposed system is to provide a secure and reliable eBook transaction system. Thus, we designed our system to satisfy the following security requirements:

- **Ownership**: It is possible to identify the owner of each eBook. It is impossible to forge or modify the owner of the eBook without knowing the private key of the owner.
- **Authentication**: It is possible to authenticate if a user is a legitimate purchaser.
- **Authorization**: It is possible to authorize if a user has a right to read a particular eBook's content. Only a legitimate purchaser who has a valid purchase transaction recorded in the blockchain has a right to read the purchased book.
- **Confidentiality of digital content**: It is impossible to obtain unencrypted and original eBook contents without knowing the book keys of the contents.
- **Non-forgeability of digital content**: It is impossible for a non-author to forge or modify eBook contents that have been already published.
- **Verifiability of digital content**: It is possible to verify the author and content validity of an eBook.
- **Readability**: The eBook contents can be restored only at the service application. It is impossible to read the contents in other ways.
- **Non-forgeability of purchase transaction**: It is infeasible to forge any eBook purchase transaction.
- **Non-repudiation of purchase transaction**: It is infeasible to repudiate previously performed purchases.
- **Verifiability of transaction**: Every eBook purchase transaction is verifiable.
- **Forbidden of piracy and illegal distribution**: Only authors have the ownership of their published eBook contents. A non-author can neither distribute nor resell the purchased book.

We assume the threat model of malicious users as follows: Malicious users can attempt to

1. upload content that does not make sense, such as a compilation of meaningless texts;
2. manipulate contents that are purchased and stored in their local devices and distribute or upload the fabricated or copied contents;
3. download contents from the book repository without valid purchase transactions;
4. stop the purchase process abnormally before the purchase process completes.

The main notations used in the proposed eBook transaction protocols are summarized in Table 3.

Table 3. Notations.

Notation	Description
U_i	The i-th user
ID_i	The ID of U_i
PW_i	The password of U_i
$< K_i^+, K_i^- >$	The public and private key pair of U_i
SA_i	The service application of U_i
MK	The master secret key of service application
$< AK_i^+, AK_i^- >$	The public and private key pair of SA_i
$<BK^+,BK^->$	The public and private key pair of the book repository
B_j	The j-th e-book of U
BID_j	The book ID of B_j
bk_j	A symmetric book key for B_j
$PKE(K^+,M)$, $PKD(K^-,C)$	A public key encryption and decryption algorithms (ECIES) with a public and private key pair $<K^+,K^->$
$E(k,M)$, $D(k,C)$	A symmetric key encryption and decryption algorithms (AES) for with a symmetric key k
$H(M)$	A cryptographic hash function (SHA256)
$Sig(K^-,M)$	A digital signature algorithm (ECDSA) for a message M with a private key K^+.
verify(*chain*,*transaction*, *block_ID*)	A blockchain algorithm to verify the validity of a transaction in the chain
add(*chain*,*transaction*)	A blockchain algorithm to add a new transaction to current blockchain

https://doi.org/10.1371/journal.pone.0228418.t003

The proposed eBook trading protocols

User registration. All eBook services are available by installing the service application. Every *SA* has the same master secret key denoted as *MK* that has been already programmed in the application in a secure manner. Let SA_i be the service application of a user U_i. The installed SA_i initially performs a user registration process, which produces U_i's ID (ID_i), password (PW_i), and public-and-private key pair (K_i^+ and K_i^-) and *SA*'s public-and-private key pair (AK_i^+ and AK_i^-); U_i chooses its own ID, password, and public-and-private key pair. The user's private information is encrypted with *SA*'s private key, and *SA*'s private key is encrypted with *MK*. Thus, it is impossible for any entity except for the *SA* itself to know the *SA*'s private key in any way. All private information is stored in U_i's local storage as encrypted. Then, user information, denoted as *user_info* (user's ID, user's public key, and *SA*'s public key), is generated as follows:

$$user\_info = "ID_i|K_i^+|AK_i^+" \tag{3}$$

user_info is broadcast with its signature $USIG_i = Sig(K_i^-, user\_info)$. All *SA*s over the network add the new user to their user lists. The user registration process is given in Fig 3:

The *SA* requires a user's password to authenticate the user, simply checking the validity of the password given whenever a user utilizes his or her *SA*.

eBook registration. Any user can upload his or her digital content through the eBook management module of *SA*. eBook registration consists of two steps: updating *B_Chain* with new book information and uploading encrypted eBook content to the book repository. Let the j-th digital content of an author U_A be B_j. SA_A first chooses a random secret key s_j and creates a book key bk_j of B_j as follows:

$$bk_j = H(s_j|MK) \tag{4}$$

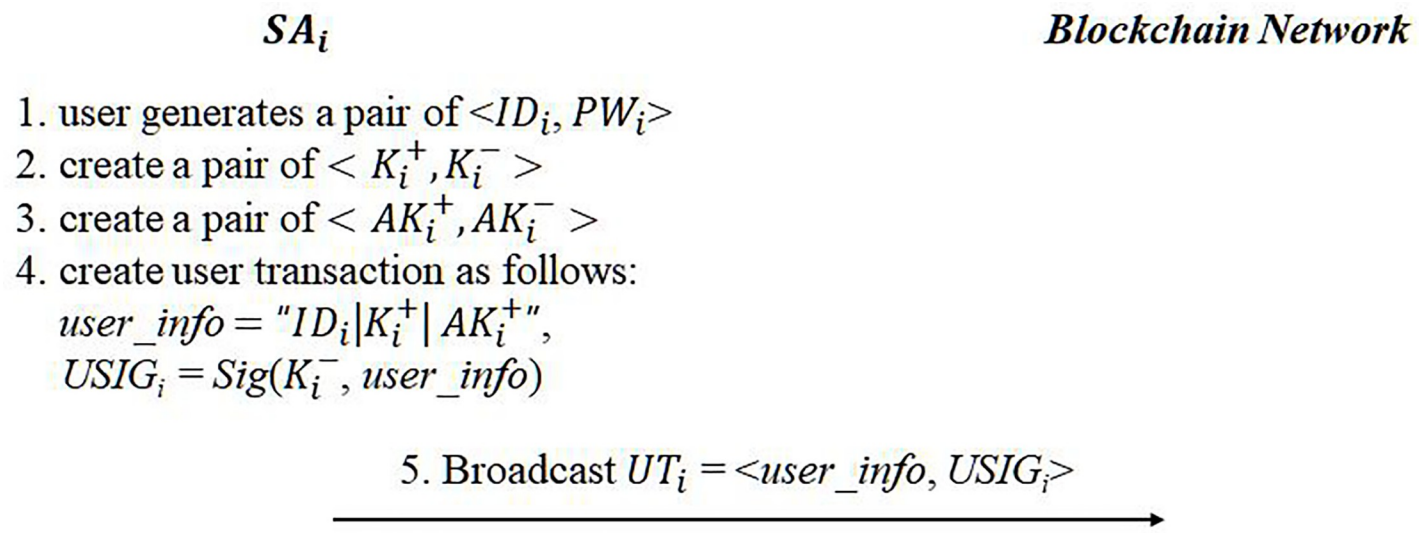

Fig 3. User registration.

https://doi.org/10.1371/journal.pone.0228418.g003

The reason the master secret key of SA_A is used to generate the book key is so that only the legitimate SA, which knows the master secret key, can produce the same book key later.

B_j is divided into n pieces of the same size except for the last piece, and these are denoted as $B_j = \{b_{j,1} | b_{j,2} | \ldots | b_{j,n}\}$; the size of the last piece is less than or equal to that of the other pieces. The number of pieces depend on the book size. SA_A generates a hash value of B_j and hash values of individual file piece $b_{j,k}$, and then produces a hash list for all file pieces, such as $H_j = H(B_j)$ and $HL_j = \{H(b_{j,1}) | H(b_{j,2}) | \ldots | H(b_{j,n})\}$. In order to protect the intellectual property of an eBook, the original content has to be encrypted; that is, there is no way for any buyer to check if the buying content is valid until the purchased is completed. A buyer can obtain only encrypted content and its corresponding book key after completing a purchase; if the revealed eBook content is meaningless or garbage, the buyer loses the purchase price. In contrast, we believe that a buyer should be able to verify an eBook's content during the purchase process. Our strategy is to allow a buyer to acquire a random piece of the book during the purchasing process to check its validity, and if that random piece is valid, the buyer continues the purchasing process.

SA_A encrypts each piece with the book key. The completely encrypted content is $CB_j = \{E(bk_j, b_{j,1}) | E(bk_j, b_{j,2}) | \ldots | E(bk_j, b_{j,n})\}$, and this generates U_A's digital signature for CB_j, which is $SB_j = Sig(K_A^-, CB_j | HL_j)$. SA_A generates book information as follows:

$$BID_j = H(book\_title | ID_A | K_A^+), \tag{5}$$

$$book\_info = "BID_j | book\_title | ID_A | K_A^+ | H_j | book\_price | SB_j | summary_j | timestamp", \tag{6}$$

where BID_j is the ID of B_j, ID_A is the author's ID, and K_A^+ is the author's public key. $summary_j$ is the summary of the book for preview. A pair of HL_j and CB_j is stored in a separate book repository, and only legal book purchasers can obtain those. The author's signature SB_j for CB_j and HL_j is used in the eBook upload to the book repository. The book repository can use the author's signature to verify that the author is uploading given content.

The author's signature for *book_info* is $BSIG_j = Sig(K_A^-, book\_info)$, and a final book registration transaction denoted as $BTRS_j$ is broadcast as follows:

$$Transaction_j = "BSIG_j | book\_info" \text{ and } BTRS_j = "H(Transaction_j) | Transaction_j" \quad (7)$$

At least, all *SA*s that generated *BTRS*s perform the proof-of-work for a block containing all *BTRS*s collected for a predefined period, and *B_Chain* is updated with the first generated block hash value. Since the updated *B_Chain* is deployed over the entire network including the book repository, every online *SA* updates their book lists to include all the new book information. Fig 4 summarized the eBook registration protocol.

As *B_Chain* is updated, SA_A uploads the encrypted contents and hash list to the book repository. Suppose that the book repository possesses all *BTRS*s recorded on current *B_Chain*. SA_A generates book-upload request message *book_upload* = "*block_ID* | *BTRS_ID* | BID_j | *timestamp*" and its signature $BUSIG = Sig(K_A^-, book\_upload)$. Also, SA_A encrypts "CB_j | HL_j" with the book repository's public key. Finally, SA_A sends the following book-upload request, *BURQST*, to the book repository:

$$BURQST = <book\_upload, BUSIG, PKE(BK^+, CB_j|HL_j)> \quad (8)$$

block_ID is the sequence number of the block that contains the *BTRS* of the uploaded eBook. *BTRS_ID* indicates the *ID* field of the *BTRS*. The book repository reveals CB_j | HL_j with its private key and finds the corresponding *BTRS* with *BTRS_ID* in the block. It obtains BID_j, K_A^+ and SB_j from *book_info* in the *BTRS* and verifies BID_j, *BUSIG* and SB_j for the given BID_j, *book_upload*, and the revealed CB_j and HL_j. If the signatures are all valid, then, the contents are stored in the repository. Fig 4 presents the eBook registration protocol.

Alternatively, the book repository can only store hash values of all *BTRS*s for saving the storage costs. In this case, SA_A sends the *BTRS* of the uploaded book along with *BURQST*. In

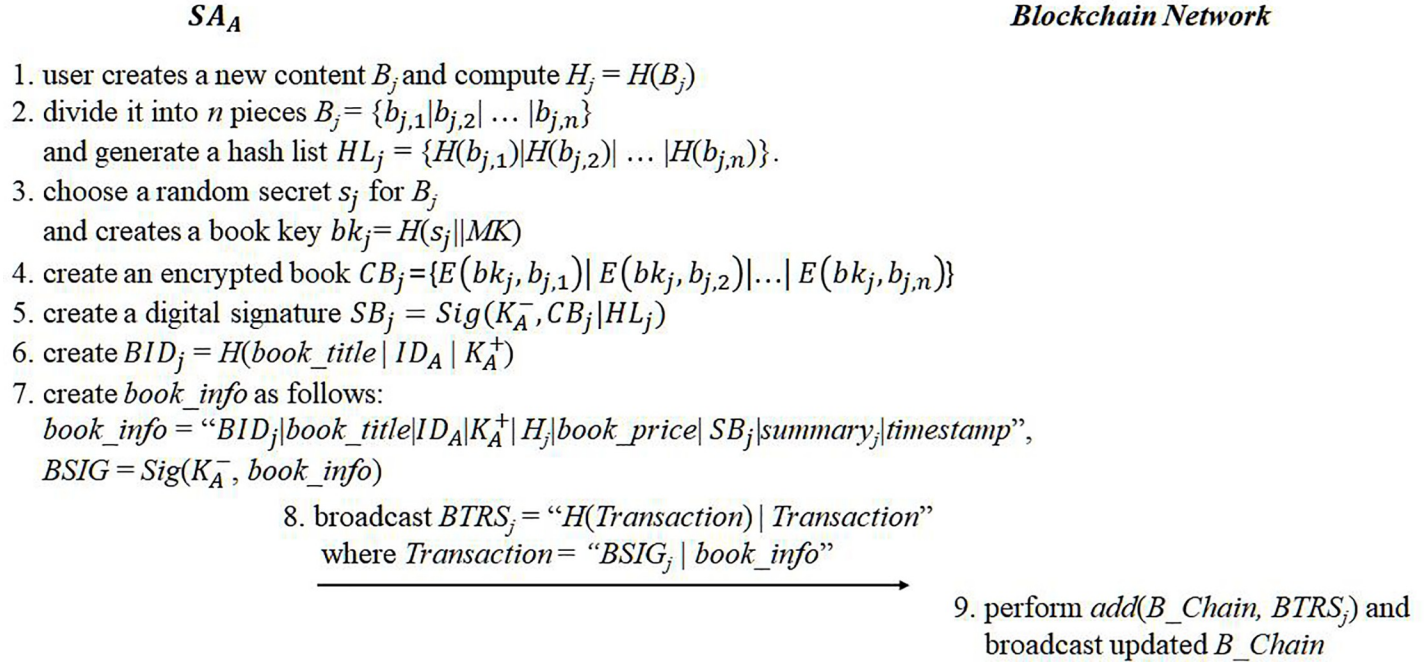

Fig 4. eBook registration.

https://doi.org/10.1371/journal.pone.0228418.g004

addition to perform the same verification process for the *BURQST*, the book repository needs to validate the *BTRS* itself. It computes the hash value of *BTRS*, the Merkle tree root hash for all *BTRS*s in the block, and the block hash. If the computed block hash is identical to the block hash of *B_Chain*, then the contents are stored in the repository. Fig 5 shows the eBook upload protocol.

eBook purchase. A user pays a fee to the author directly for the right to read an eBook. Thus, the main goal of the purchase process is to securely provide a buyer with a valid book key through a validated eBook purchase contract between the author and the buyer. Before making the agreement, the buyer should be able to validate both the purchased book's content and the received book key. Also, the author must receive the buyer's confirmation that the buyer has obtained a valid book key. To achieve this, the two sides perform the following eBook purchasing process interactively.

Let SA_R and SA_A be the service application for a book reader and an author, respectively. SA_R first generates a contract request message *request* as follows:

$$CID = H(BID_j|ID_A|ID_R|timestamp), \tag{9}$$

$$request = "CID|BID_j|ID_A|ID_R|K_R^+|AK_R^+|book\_price|balance_R|rindex|timestamp", \tag{10}$$

where *CID* is a new contract ID, AK_R^+ is SA_R's public key, $balance_R$ is U_R's current coin balance, and *rindex* is a random number in the interval [1, n_{BP}], where n_{BP} is the total number of pieces of the book. *rindex* is required to validate the content of a book for purchase. The reader's private key digitally signs the request message as $RSIG_R = Sig(K_R^-, request)$. The final contract request message $CRQST_R = <request, RSIG_R>$ is then broadcast.

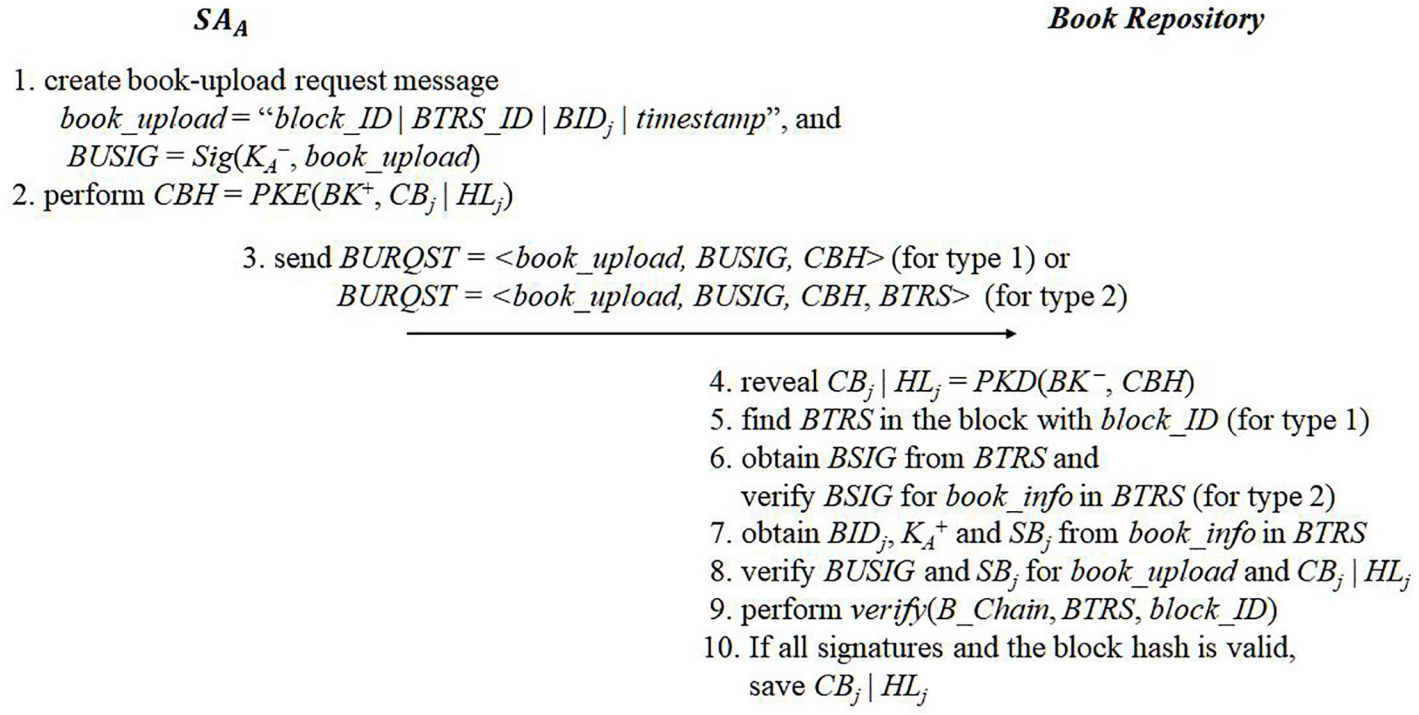

Fig 5. eBook upload.

https://doi.org/10.1371/journal.pone.0228418.g005

Second, SA_A responds to SA_R with a contract message and a book key, and the book repository responds to SA_R with the encrypted book piece and its hash value corresponding to *rindex* specified in *request*. SA_A generates a contract message and its digital signature. SA_A replies with a message that includes an encrypted secret book key, the contract message, and its signature. The contract message, denoted as *contract*, is defined as follows:

$$contract = "CID|BID_j|ID_A|K_A^+|ID_R|K_R^+|AK_R^+|book\_price|nbalance_A|nbalance_B|timestamp" \quad (11)$$

$nbalance_A$ is the updated balance of U_A after adding the book price to U_A's current balance, and $nbalance_R$ is the updated balance of U_R after the book price is deducted from the U_R's current balance. The author's private key digitally signs the contract message as $CSIG_A = Sig(K_A^-, contract)$. SA_A encrypts a partial book key s_j with SA_R's public key as follows:

$$ck_j = PKE(AK_R^+, s_j) \quad (12)$$

This is to allow for only a legal SA_R to produce a valid book key. The final contract reply message $CRPY_j = < ck_j, contract, CSIG_A >$ is then broadcast.

The book repository chooses the encrypted book piece and hash value corresponding to *rindex* specified in the request message and encrypts it with SA_R's public key as well. Note that SA_R's public key is specified in *request*. The repository responds to SA_R with $cb_{j,rindex}$ as follows:

$$cb_{j,rindex} = PKE(AK_R^+, E(bk_j, b_{j,rindex})|HL_{j,rindex}) \quad (13)$$

Lastly, SA_R validates the given book key and the eBook content. If both the key and the content are valid, then SA_R broadcasts a confirm message digitally signed by the reader's private key; otherwise, SA_R generates a fail message signed by the reader's key. SA_R obtains the partial book key by decrypting ck_j with SA_R's private key as $s_j = PKD(AK_R^-, ck_j)$. Then, SA_R produces the original book key using its master secret key as $bk_j = H(s_j \mid MK)$. Also, SA_R decrypts $cb_{j,rindex}$ with its private key and obtains the encrypted book piece $E(bk_j, b_{j,rindex})$ and the corresponding hash value $HL_{j,rindex}$. SA_R decrypts $E(bk_j, b_{j,rindex})$ with its generated book key bk_j and produces a hash value of $H(b_{j,rindex})$. If the produced hash value is identical to the given value, then the book key is proven correct. In addition, it is proven that the contents stored in the book repository are the original contents that the author created and uploaded (that is, the content was not modified).

SA_R shows a fragment of the decrypted book piece $b_{j,rindex}$ to U_R. U_R checks if the decrypted content makes sense, that is, if it consists of meaningful text. If both the key and the content are valid, SA_R generates a digital signature for *contract* $CSIG_R = Sig(K_R^-, contract)$ and generates and broadcasts a final purchase transaction, *CTRS*, as follows:

$$Transaction = "CSIG_R|CSIG_A|contract" \text{ and } CTRS = "H(Transaction)|Transaction" \quad (14)$$

Otherwise, SA_R broadcasts a fail message $<MSG$ = "verification failed", $Sig(K_R^-, MSG)>$ and the purchase is terminated. For a valid transaction, SA_A just generates a block containing *CTRS* and adds it to *C_Chain*. *C_Chain* is broadcast to all users including the book repository, and SA_R and SA_A update their users' coins. The book repository does not need to store all *CTRS*s but only keeps the transaction hash values, block nonce and the block hash. Fig 6 summarizes the eBook purchase protocol.

eBook download and storage. Once the contract has been completed successfully, SA_R can download the encrypted eBook pieces from the book repository. SA_R first generates a book request message with its transaction, and if the transaction is valid, then the repository responds with the encrypted book pieces. The book request message is constructed as

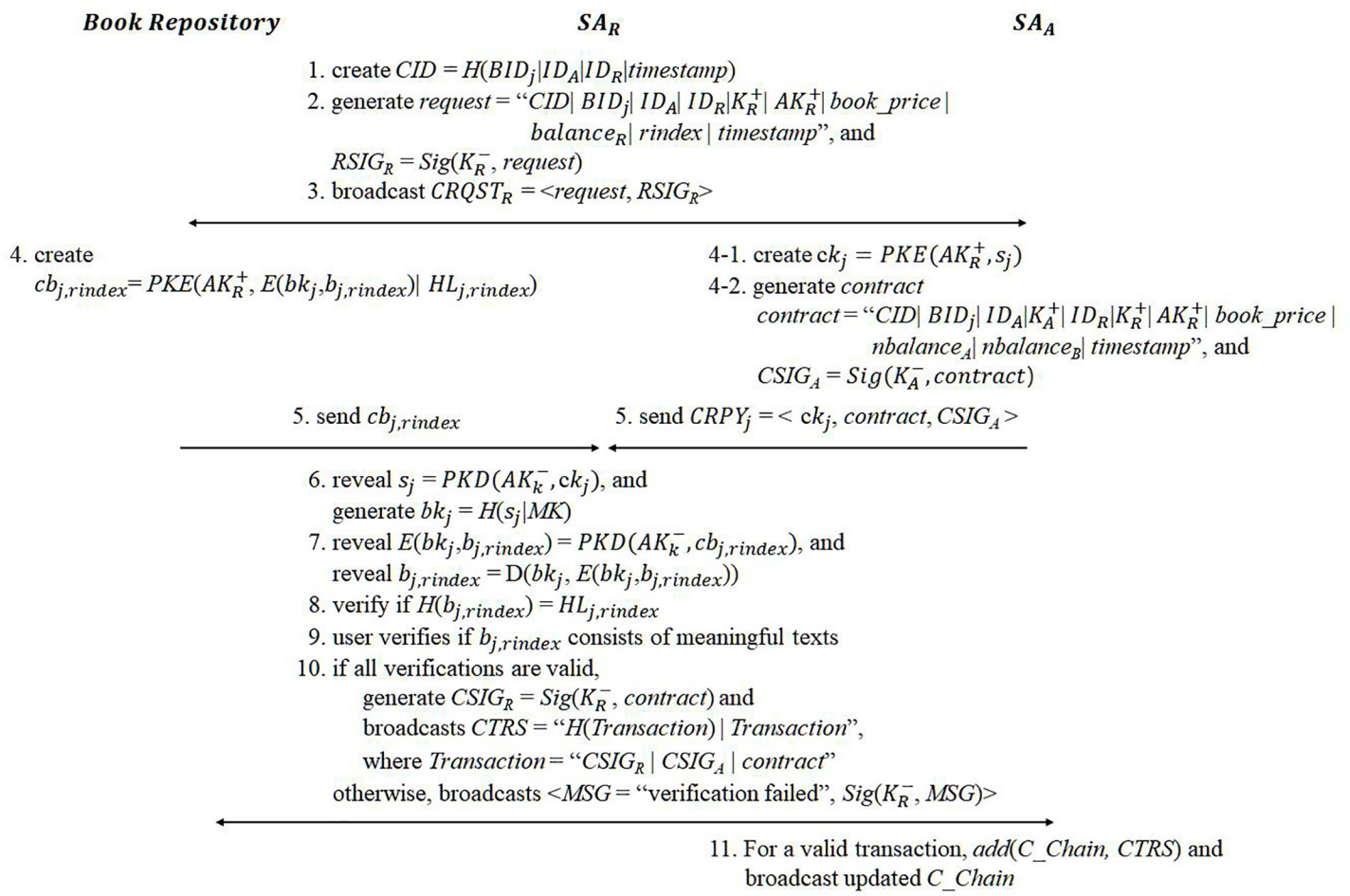

Fig 6. eBook purchase transaction.

https://doi.org/10.1371/journal.pone.0228418.g006

$book\_request$ = "$block\_ID\ |\ CTRS\_ID\ |\ BID_j\ |\ timestamp$" and its signature is generated as $BRSIG = Sig(AK_R^-, book\_request)$. A final book request, $BRQST$, is sent to the book repository:

$$BRQST = <book\_request, BRSIG, CTRS> \quad (15)$$

$block\_ID$ is the sequence number of the block containing $CTRS$, and $CTRS\_ID$ indicates the ID of $CTRS$. The book repository obtains AK_R^+, K_A^+ and K_R^+ from $contract$ in $CTRS$. It firstly verifies the validity of $BRSIG$ with AK_R^+ and verifies two signatures $CSIG_R$ and $CSIG_A$ for $contract$ with K_A^+ and K_R^+. If the signatures are all valid, then it computes the hash value of $CTRS$ and its block hash. If the computed block hash is identical to the block hash in the C_Chain, then the transaction is validated. Then, the repository replies with CB_j encrypted with AK_R^+ as $PKE(AK_R^+, CB_j)$. Because a valid C_Chain is required to download the eBook, it is impossible to obtain the original eBook content without completing the purchase transaction, even though the book key has been obtained. The eBook download protocol is given in Fig 7:

Once the eBook has been downloaded, SA_R decrypts the original content of the book and re-encrypts it with SA_R's randomly chosen secret key, and then it stores the book in the reader's local storage. Thus, even for the same eBook, each reader receives a differently encrypted version of the book for local storage. Additionally, each encrypted version can be decrypted only by SA. The detailed protocol is given in Fig 8:

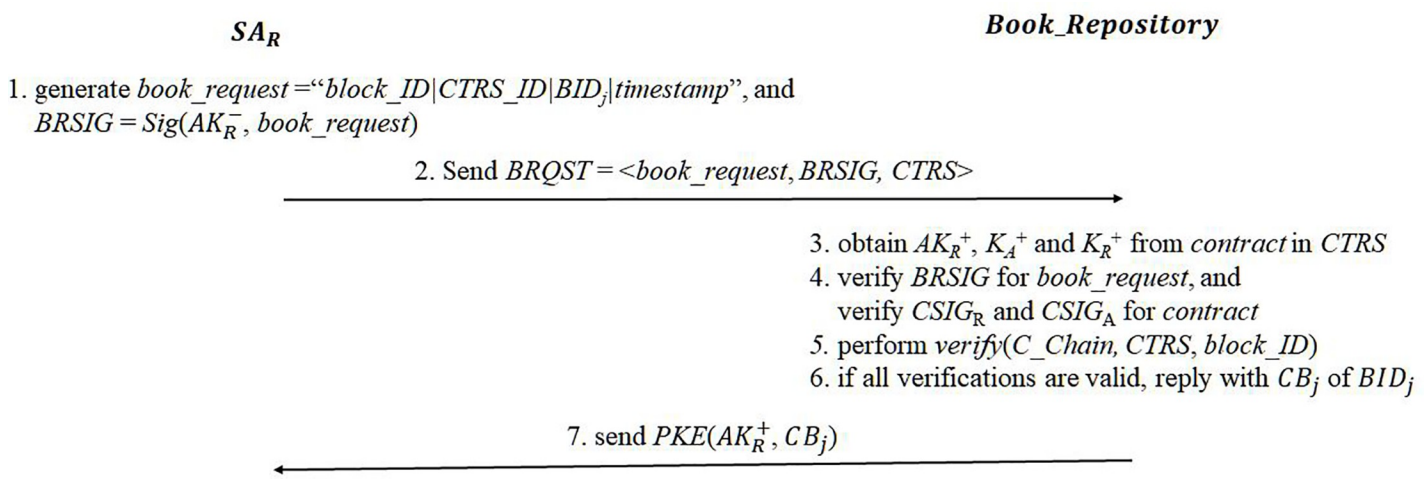

Fig 7. eBook download.

https://doi.org/10.1371/journal.pone.0228418.g007

Users can read books they have purchased legally through their *SA*. Whenever an eBook read request occurs, *SA* reveals the book key $n\_bk = PKD(AK_R^-, c\_bk)$, reveals the book $B_j = D(n\_bk, C_j)$, and shows the book through the *SA* viewing module. Because the purchased content is encrypted with a new book key and also with the *SA*'s public key, the user cannot obtain the original and unencrypted content from the local device. Because the encrypted book key can be restored only by the *SA*'s private key, the other *SA*s cannot restore the encrypted contents without knowing the *SA*'s private key, even if the user distributed the encrypted content to others.

Additional remarks

We explain further some distinct features of the proposed system by answering the following three questions.

(1) Why is the master secret key of SA required?

Every *SA* shares the same master key, which is securely programmed but is not stored in anywhere. The master secret key is mainly used to generate the book key for a segment of

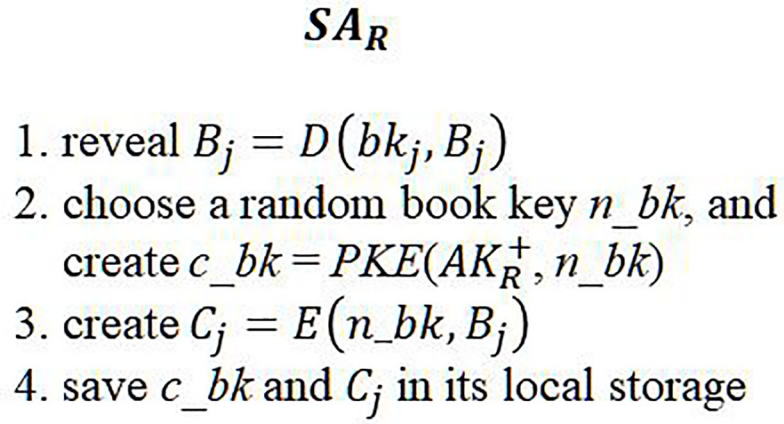

Fig 8. Re-encryption and save eBook.

https://doi.org/10.1371/journal.pone.0228418.g008

eBook content. When a new book is created, *SA* first chooses a random secret and the book key is generated as a hash value of a concatenated string with both the random secret and the *SA*'s master secret, such as *H*(a random secret | *SA*'s master secret). When a user wants to buy the book, the author sends not the book key but only an encrypted random secret. Then, the purchaser should generate the same book key with the given secret and the master secret of the purchaser's *SA*. This is to authenticate if the purchasing object is a valid service application. In other words, this is to prevent a malicious purchaser from obtaining the original contents using a fake *SA*.

(2) Why is the book repository necessary?

We have assumed every registered eBook's content is uploaded to the book repository, which acts as a content server; the original purpose of the repository is to increase the efficiency of downloading content. Suppose that every author keeps all contents in his or her local storage without the book repository; an author receives every request to buy his or her books, and every buyer should receive the books from the author, and under these circumstances, the overload to the author's network could be heavy. The book repository acts as a kind of cloud server that can handle many requests and downloading tasks at the same time, and it can be implemented in a decentralized way. The book repository is not a trusted party, that is, it takes no responsibility for the security of stored contents; it has only a control logic, which can validate basic cryptographic protocols such as digital signatures and blockchain hash values. The book repository is necessary for practical use of the proposed system.

(3) Why does a purchaser need to receive a random book piece from the book repository?

The proposed purchasing protocol is performed by three parties: a buyer, an author, and the book repository. To validate the book for purchase, the buyer requests a randomly selected book piece, then the book repository responds to the buyer with the corresponding encrypted piece and its hash value: This process is the most crucial part of the proposed system. After the purchase has been completed, the buyer downloads the entire eBook contents from the book repository. In other words, from the buyer's point of view, it is necessary to verify not only the content for purchase but also the integrity of the contents stored in the book repository; that is, the buyer should be able to verify if the stored contents are the original contents that the author uploaded. Our protocol resolves all of these challenges. First, if the random book piece the book repository gives is restored correctly, that is, if the hash value of the restored content is the same as the given hash value, the buyer can validate the correctness of the given book key as well as the integrity of the content stored in the repository because only the author can create the hash value of each eBook piece; if the restored content is meaningful, then the content is validated. What if not the book repository but the author provided the buyer the random piece during the purchase process? The buyer can validate the given book key for the encrypted book piece given from the author, but, the buyer cannot be sure that the other contents stored in the repository are all valid. What if the buyer receives the encrypted book piece from the author and receives the hash value from the book repository? This is the same as the previous situation: The buyer still cannot be sure of the association of the encrypted content stored in the book repository with the hash. Therefore, obtaining a random piece of eBook content from the repository is necessary.

(4) What if a user uploads plagiarized content?

Firstly, a user cannot obtain the original text file of the purchased eBook content without knowing the master secret key of the service application. That is, making a copy of the original text file of the purchased eBook content is impossible. A user, however, can read the purchased book, create new plagiarized eBook content, and publish it as new eBook content. If the plagiarized eBook is exactly same as the existing eBook content, since the hash value of the existing

creation is the same as that of the plagiarized creation, it is possible to determine whether plagiarism is performed by comparing the hash values. Plagiarism detection in this way, however, is very inefficient and impractical. Even with a slight change in the text, the hash value is different, so it is not possible to detect plagiarism unless it is a file copy. The proposed system does not automatically detect plagiarized content and does not prevent users from uploading plagiarized content. Further research is needed to solve the plagiarism detection. Typically, user review and reputation systems can be applied to the proposed system.

Evaluation and analysis

Security analysis

In this section, we analyze how the proposed eBook transaction system satisfies the suggested security requirements.

(1) Ownership, Authentication, and Authorization. First, when a new eBook is created, book information is created including book title and id, author's id and public key, produced, and digitally signed by the author; then the information is broadcast to all users on the network and recorded on the blockchain named as *B_Chain*. Thus, any user can authenticate the owner of each eBook publicly.

Second, any buyer who performed the purchase process correctly obtains a valid purchase transaction and its blockchain containing the transaction block hash. Because the transaction includes signatures generated by both the buyer and the author, a legitimate purchaser can be authenticated by validating the transaction and the blockchain.

Lastly, the only user who possesses both a valid book key for the purchased eBook and a valid contract chain containing the purchase transaction of the book can read the eBook; the book key is required to decrypt the purchased eBook, and the contract chain is necessary to obtain the remainder of the encrypted eBook contents from the book repository. The purchase process consists of two steps: (1) obtaining a book key for an eBook and (2) performing the agreement to the eBook purchase between the buyer and the author. A malicious buyer may perform only the first step and stop the rest of the contract process; then, the contract will not be completed so that a valid contract chain containing the transaction will not be produced. Even though the buyer has received a valid book key, it is not possible to download all the rest of encrypted eBook contents from the book repository without a valid transaction and its blockchain. Therefore, only the legitimate purchaser who has completed the purchase process normally has a right to read the purchased book.

(2) Confidentiality of digital content. Basically, every eBook has been encrypted with its book key and stored in the book repository. Every eBook is communicated over the network as encrypted with a receiver's public key. The book key is randomly generated for each eBook based on the master key of the service application. In addition, the book key is delivered to a buyer as encrypted with the public key of the buyer's *SA*; thus, only a valid *SA* that knows the *SA* master key can reveal the book key. On top of it, the purchased eBook is re-encrypted for a new key chosen by the buyer's *SA* and saved in the buyer's local storage; thus, the purchased eBook can be opened only by the buyer's *SA*. Also, the new book key is also encrypted with the *SA*'s public key. That is, all information related to the eBook is stored as encrypted so that it is impossible for even a legal buyer to obtain the original unencrypted content without knowing the *SA* master key.

(3) Non-forgeability of digital content. As mentioned above, because every eBook is released as encrypted with its own book key, accessing and forging the original content is impossible without knowing the corresponding key. Forging the contents stored in the book repository is also infeasible because eBook uploading requires the corresponding book

information digitally signed by the book. Thus, it is impossible to forge already released eBook contents without knowing the private key of the author.

(4) Verifiability of digital content. Each eBook is released with the hash value of the original content, and the author digitally signs the book information; with the signature, the buyer can verify the author of the book. During the purchase protocol, the buyer's *SA* generates the purchased book's book key and decrypts a randomly selected book piece, allowing the buyer to directly verify if restored content is meaningful. When a buyer has downloaded all encrypted book pieces from the book repository, the buyer can verify the integrity of the downloaded contents by comparing the hash value of the revealed content with the published hash value of the original content.

(5) Readability. The purchased eBook is only readable by the buyer's *SA* because only that *SA* can decrypt the corresponding book key with its private key and present the original content. The purchased contents are stored in the buyer's local storage, so users can read the books any time.

(6) Non-forgeability of transaction. The eBook purchase is an interactive process between a buyer and an author, and the process requires consensus. If and only if both sides generate valid digital signatures about the eBook purchase contract is the contract added to the contract chain. According to the basic nature of the blockchain technology, a valid contract chain can be produced if and only if a valid proof of work about a block containing the contract is generated. Thus, forging a transaction requires forging digital signatures of both sides without knowing the private keys of either side and forging the blockchain as well, whereas this is impossible due to the security of the cryptographic digital signature and of the Blockchain. Therefore, our eBook transaction is non-forgeable.

(7) Verifiability of transaction. As mentioned above, the eBook contract contains the digital signatures of both a buyer and an author, and the contract is permanently stored in the contract chain. Thus, the contract can be always verified with the public keys of both sides and current contract chain due to the nature of the blockchain.

(8) Forbidden of illegal distribution. Ultimately, users cannot obtain the original contents of eBooks that they did not create from their local storages because users cannot obtain valid book keys of the purchased eBooks. Therefore, it is impossible to redistribute the original contents of purchased eBooks. In other cases, the encrypted contents stored in the user's local storage may be redistributed, but the book key cannot be restored because the key is encrypted with the private key of the user's *SA*. The encrypted content can be decrypted only by the *SA* of the original purchaser, and the redistribution is meaningless because other *SA*s cannot decrypt the content normally.

Simulated performance analysis

In this section, we examine the simulated performance of the proposed eBook transaction system. Our model should be serviced in real-time, so we analyze the average execution time for the following seven main actions: eBook generation, eBook registration transaction (*BTRS*) broadcasting, eBook upload to the book repository, eBook purchase transaction (*CTRS*) generation, blockchain update, eBook download from the book repository, eBook resave. Five experimenters randomly repeated the eBook registering and purchasing process. The experimenters performed the registration and purchase process at least 10 times for each eBook size. Our experimental results show the average value of all execution times performed by the experimenters for each main action. In particular, the eBook upload and the eBook download require additionally transaction verification with blockchain. In other words, *BTRS* verification using *B_Chain* and *CTRS* verification using *C_Chain* are required before the eBook

Table 4. Simulation environment parameters.

Parameter	Values
OS	Windows 10
RAM	16 GB
CPU	Intel i7-6700
The number of nodes	5
Development tool	Java
Public key encryption / decryption	ECIES
Digital signature	ECDSA
Symmetric key encryption / decryption	AES 256
Message hash algorithm	SHA 256
User information type	JSON format
The original eBook type	Text file (.txt)
eBook information type	JSON format
eBook file size	50KB, 100KB, 500KB, 1MB, 3MB, 5MB, 10MB
The level of difficulty	3, 4, 5
Blockchain and digital coin	Self-developed coin and blockchain mechanism

https://doi.org/10.1371/journal.pone.0228418.t004

upload and the eBook download, respectively. Thus, each transaction verification time will be analyzed separately. We describe first the experimental environment including user information and eBook information, and analyze our experimental results in detail. The detailed experimental environment for the simulation is described in Table 4.

Our system begins with the installation of the proposed *SA*, which registers a new user, creates new user information, and broadcasts it to existing users. The *SA* obtains a current user list from a predefined genesis server; the user information contains user's id (*uid*), user's public key (*upk*), user's network ip and port (*ip* and *port*), and the public key of the user's SA (*spk*). The user information is produced as a JSON format as shown in Table 5 and is broadcast to all existing users. Then, users update their current user lists.

Whenever a user creates new eBook content, the book registration transaction containing the book information is broadcast to all users on the network, and encrypted eBook pieces are uploaded to the book repository. The book information shown in Table 6 contains book id (*bid*), book title (*title*), author's id (*uid*), author's public key (*upk*), book price in coins (*price*), signature for encrypted book pieces and hash list (*signatureStr*), book summary (*summary*), book size in bytes (*size*), the total number of book pieces (*numOfPices*), and timestamp. Each encrypted book piece is filed in bytes, and its file name consists of its *bid* and its piece number.

An eBook is divided into small book pieces, and the number of book pieces is determined according to the book size as shown in Table 7.

The main processes of our system are (1) eBook registration, (2) eBook purchase, and (3) eBook download and storage. Registering an eBook consists of generating a new eBook, updating *B_Chain* with a new book registration transaction, and uploading the eBook to the book

Table 5. An example of user information in JSON format.

{"uid":"soyoung",
"upk":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAExxTLAUMcO1NbcXvXY
tga0UxOQ3kf18B9J8h65OirAOTz5Cr3xpj00Bw23QMS0jk6",
"port":4000,
"ip":"000.000.000.000",
"spk":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAE9Iayc7zj+q5tv8OxV9WcB
LdAWAwUC0ZOxsbxMKojti9UOin8DkIJB0r0AOkYgCTJ"}

https://doi.org/10.1371/journal.pone.0228418.t005

Table 6. An example of eBook Information in JSON format.

{"bid":"5d84cd965061bb352e88858463dac17642836fba408f2881f67977f1ea03762a",
"title":"p_50_2",
"uid":"Alice",
"upk":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEFDJG1IAURDXX2460oHP8kjc2RgE1yviprY3x4cr/GE52kfBxkwhmVi1nXYVXRnC/",
"price":2.0,
"signatureStr": "MDQCGFiiTGH1bOXfmc182WqcOzMMliWWI756hQIYUNarqqI65zQ/wdABhPpsNZgoeFnDpdzM",
"summary":"This is a test book file.",
"size":51253,
"numOfPieces":103,
"timestamp":"2019-11-24 13:55:37.213"}

https://doi.org/10.1371/journal.pone.0228418.t006

Table 7. The number of file pieces.

	50KB	100KB	500KB	1MB	3MB	5MB	10MB
The number of pieces	103	103	257	513	615	1025	2050

https://doi.org/10.1371/journal.pone.0228418.t007

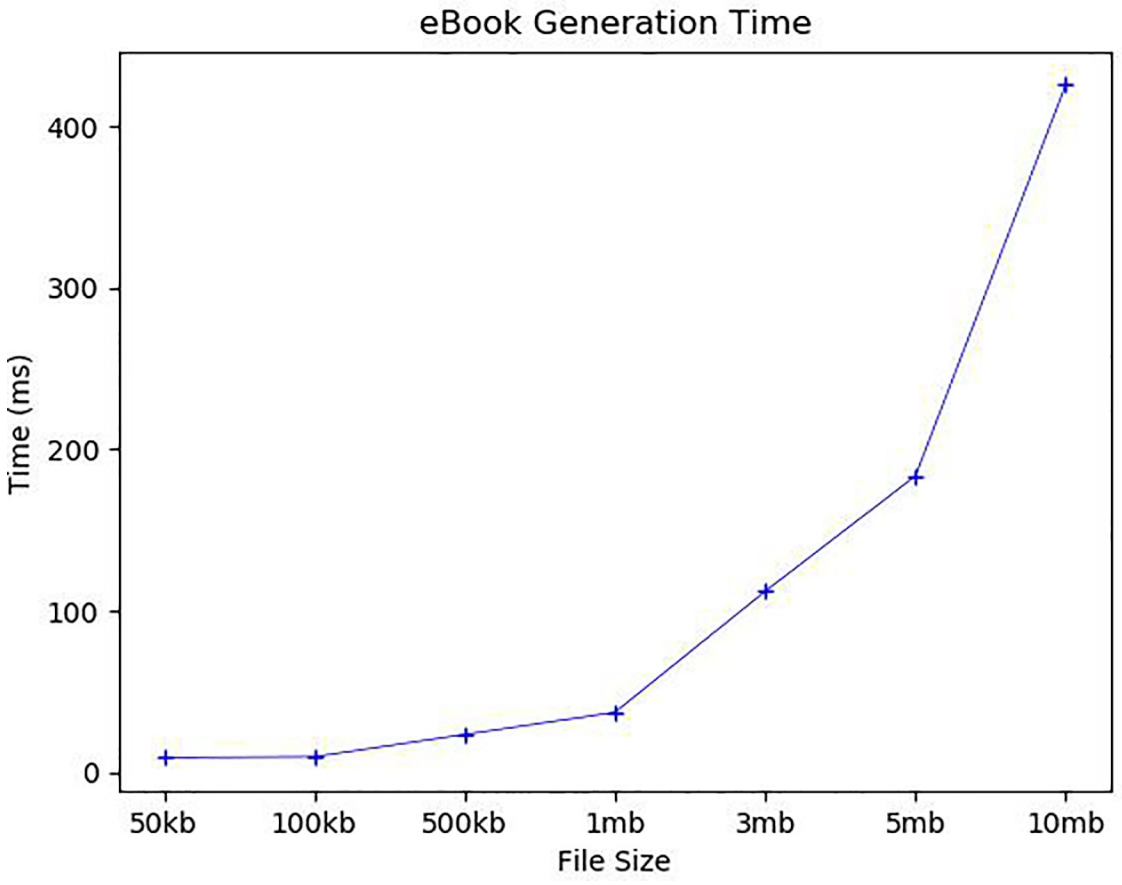

Fig 9. eBook generation time.

https://doi.org/10.1371/journal.pone.0228418.g009

repository. The eBook purchase is divided into two steps: a new eBook purchase transaction and *C_Chain* update to include the new transaction. The first step entails validating the eBook and the mutual transaction between a buyer and an author; it involves the buyer reading and

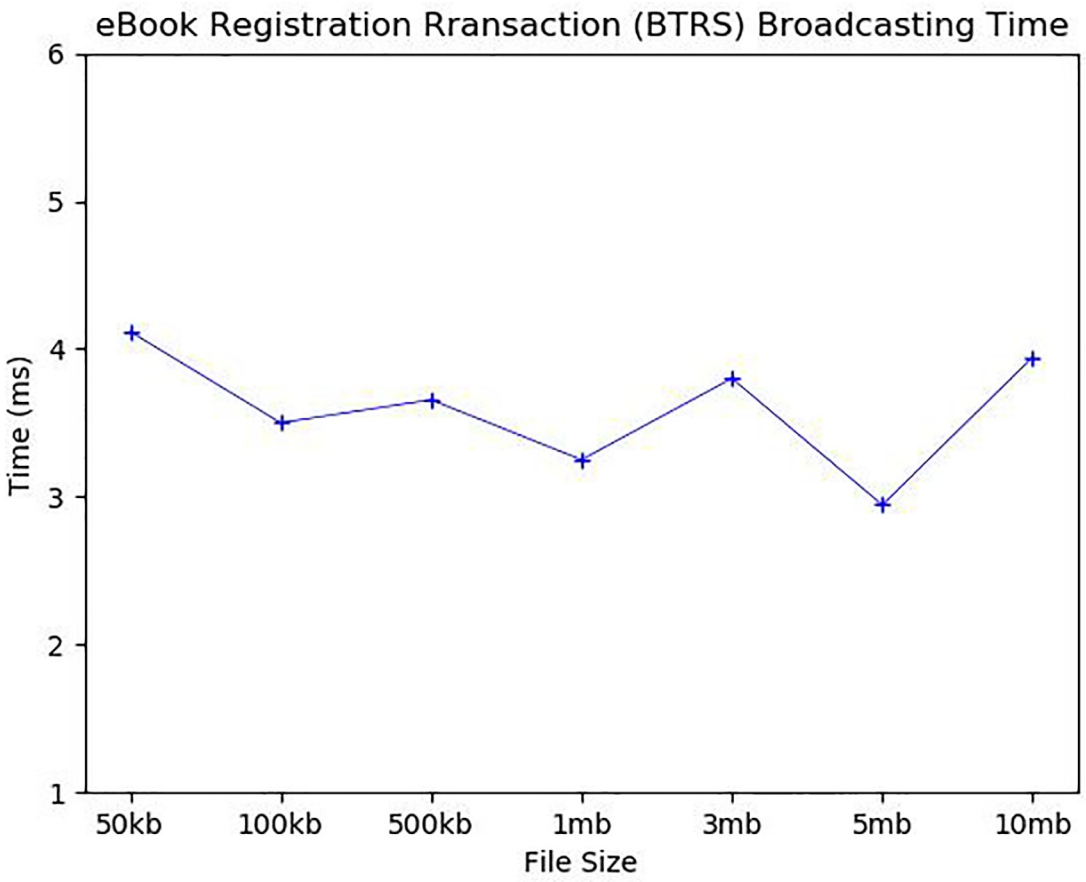

Fig 10. eBook registration transaction (*BTRS*) broadcasting time.

https://doi.org/10.1371/journal.pone.0228418.g010

confirming the revealed content. If the first step is performed correctly, the second step takes place: *C_Chain* is updated with the addition of a new block that contains the transaction. In our simulation, the buyer finally completes and sends the eBook purchase transaction to the author, and then the author creates and replies with a new block by mining the proof-of-work of the block for practicality and efficiency. As soon as the eBook purchase is complete, the eBook download is performed automatically. More specifically, the execution time of each detailed step is determined as follows:

- **eBook generation time** = load contents from text file + generate *bid* + generate book key + generate encrypted book pieces + generate book registration transaction (*BTRS*)
- ***BTRS* broadcasting time** = send book information JSON file to a single user
- ***B_Chain* and *C_Chain* update time** = create block + mine block
- **eBook upload time** = generated book upload request + send request and list of encrypted book pieces + verify request with *B_Chain* + save book pieces in files
- **eBook purchase transaction (*CTRS*) generation time** = generate book purchase request + generate and verify signature + send request + encrypt book key + generate contract message + generate and verify signature + decrypt book secret and generate book key + decrypt random book piece + confirm the decrypted book piece + generate *CTRS*

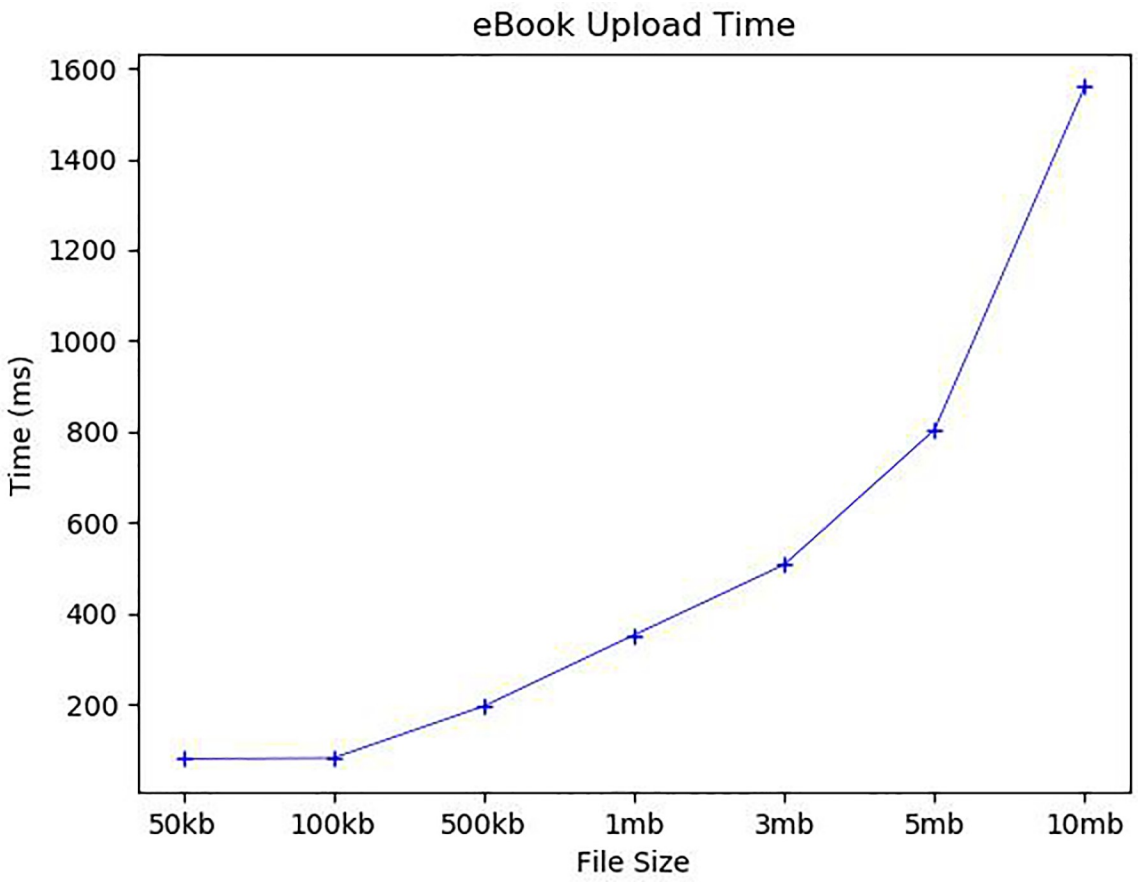

Fig 11. eBook upload time.

https://doi.org/10.1371/journal.pone.0228418.g011

- **eBook download time** = generate book download request + send book request + verify request with *C_Chain* + download whole pieces of eBook
- **eBook resave time** = decrypt eBook + generate new book key + encrypt the book with new book key + save the encrypted book and book key in local storage

We now analyze the average execution time of each detailed step. Fig 9 first shows the average time to generate an eBook: It took 9.1ms and 9.7ms to generate 50 and 100 kb eBooks, 23.5ms for eBooks of 500 kb, 37.1ms for 1 mb, 112.1ms for 3 mb, 183.6ms for 5 mb, and 425.6ms for 10 mb. Since the most time-consuming task in the eBook generation is to create encrypted book pieces, the eBook generation time increases proportionally to the number of book pieces and the size of each book piece. The average *BTRS* broadcasting time to a single user is about 3.5ms as shown in Fig 10. Because *BTRS* contains only the book information regardless of eBook size, the broadcasting time is very dependent on the network environment. We conducted our simulation on the same local network, so the broadcasting time was very short. *B_Chain* for broadcasted *BTRSs* is updated periodically. The *B_Chain* update time will be analyzed along with *C_Chain* update time later.

Fig 11 shows the average time to upload the eBook to the book repository. The encrypted eBook file pieces and the hash list of the file pieces are delivered to the book repository as encrypted with the book repository's public key. The length of hash list increases in proportional to the number of file pieces. The eBook upload includes verifying the upload request

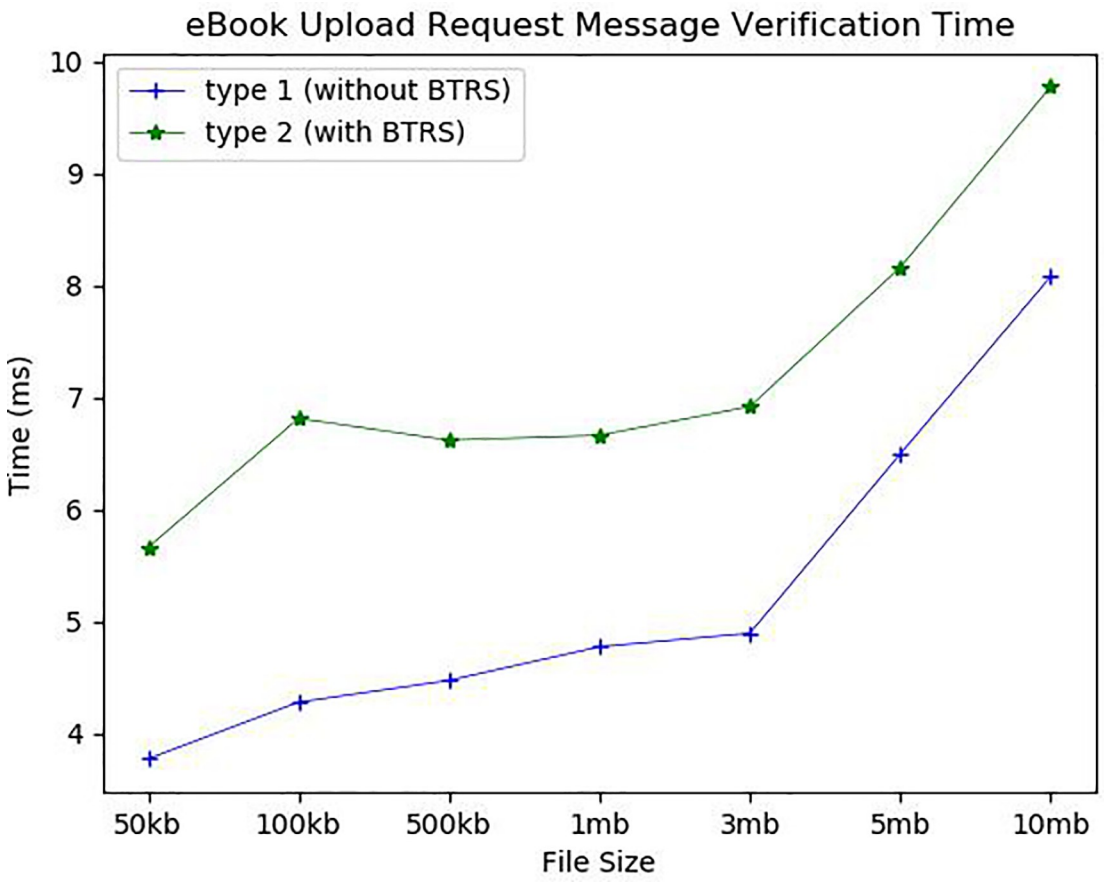

Fig 12. eBook upload request message verification time.

https://doi.org/10.1371/journal.pone.0228418.g012

message along with uploading the encrypted file pieces. The upload request verification consists of decrypting the delivered message and validating the upload request message with *B_Chain*. If the verification is valid, each encrypted file piece is saved as a separate file in the book repository. Fig 12 shows the average eBook upload request verification time for two types: Type 1 (without *BTRS*) assumes for the book repository to store all *BTRS*s recorded in *B_Chain* but Type 2 (with *BTRS*) assumes to keep only the hash of every *BTRS*. Thus, for type 2, *BTRS* of the uploaded eBook should be delivered to the book repository and additional verification for the given *BTRS* itself is necessary. For Type 1, it took around 3.7ms for 50 kb, 4.3ms for 100 kb, 4.5ms for 500 kb, 4.7ms for 1mb, 4.9ms for 3mb, 6.6ms for 5mb and 8ms for 10 mb. For Type 2, it took around 5.6ms for 50 kb, 6.8ms for 100 kb, 6.6ms for 500 kb and 1mb, 6.9ms for 3 mb, 8.1ms for 5 mb and 9.7ms for 10 mb. The computational overhead for Type 2 verification is around 3ms. For the eBook upload, it took about 81.25ms and 83ms to upload 50 and 100 kb, 196.8ms for 500 kb, 353.25ms for 1 mb, 508.2ms for 3 mb, 804.5ms for 5 mb, and 1558.75ms for 10 mb. The eBook request verification time and eBook upload time for 5 mb and 10 mb files increased more significantly than others because the number of file pieces and eBook size nearly doubled than 3 mb and 5 mb eBooks, respectively. In addition, the longer hash list was added to those files. Our experimental results show that the eBook upload time is mainly affected by the number of file pieces and the size of each file piece. We will continue further research to find the optimal number of file pieces that ensure both the eBook safety and communication efficiency.

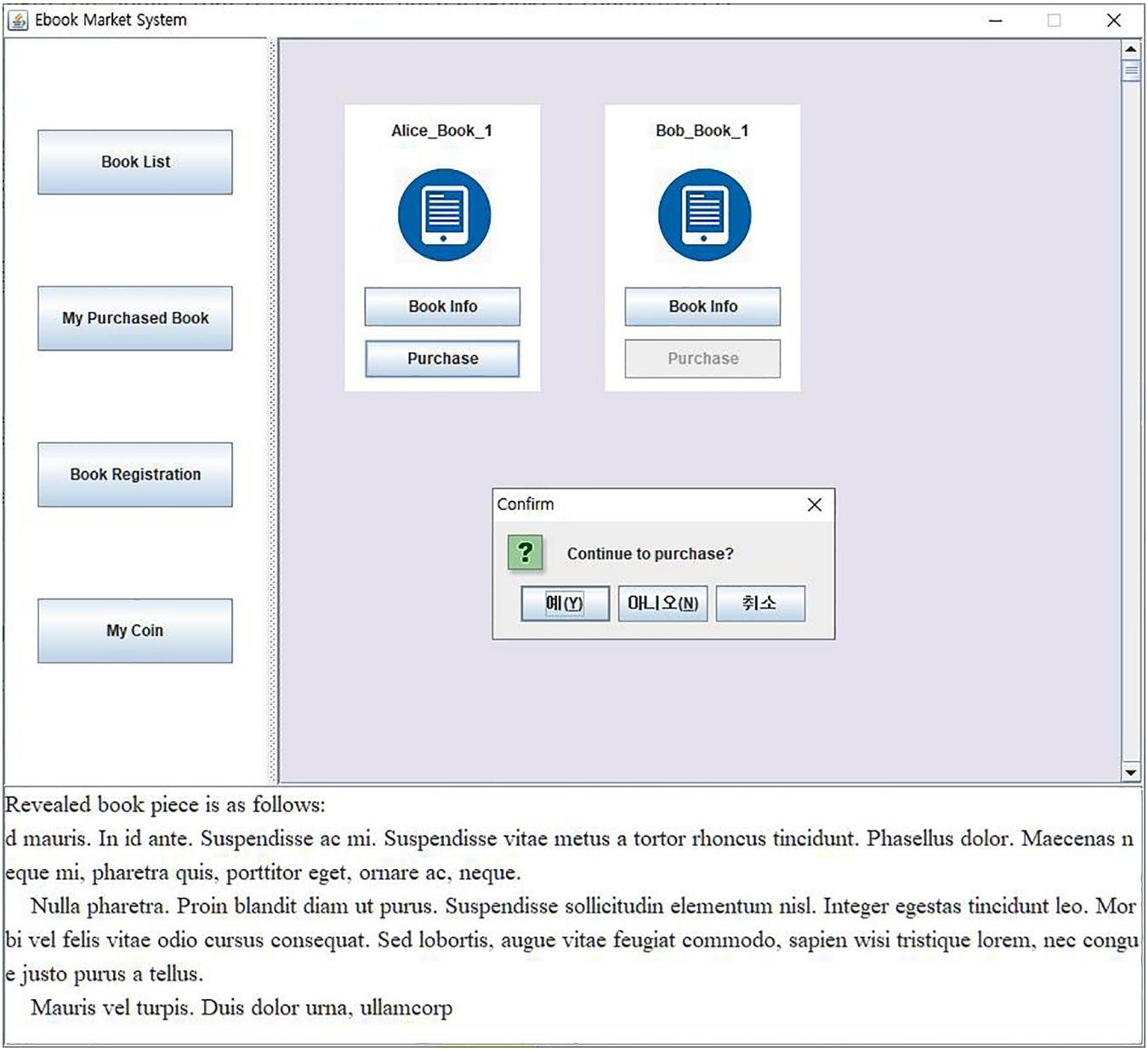

Fig 13. User confirmation of the decrypted book piece.

https://doi.org/10.1371/journal.pone.0228418.g013

As mentioned above, eBook purchase consists of generating an eBook transaction and updating a corresponding blockchain. The most time-consuming task in generating the transaction is the buyer's content verification, during which the buyer reads partially revealed contents, verify if the content makes sense, and then confirms purchase by clicking "yes" in the confirmation dialog window. Fig 13 shows the confirmation performed by *SA*. The revealed content is presented at the bottom text filed of *SA*. Thus, the transaction generation time depends heavily on the buyer's response behavior. In our experiments, the experimental participants usually responded in about 2 seconds as shown in Fig 14. Once the buyer and the author have made an agreement on their transaction, a valid transaction is completed, and then, the blockchain is updated. A new block is generated that contains the new transaction,

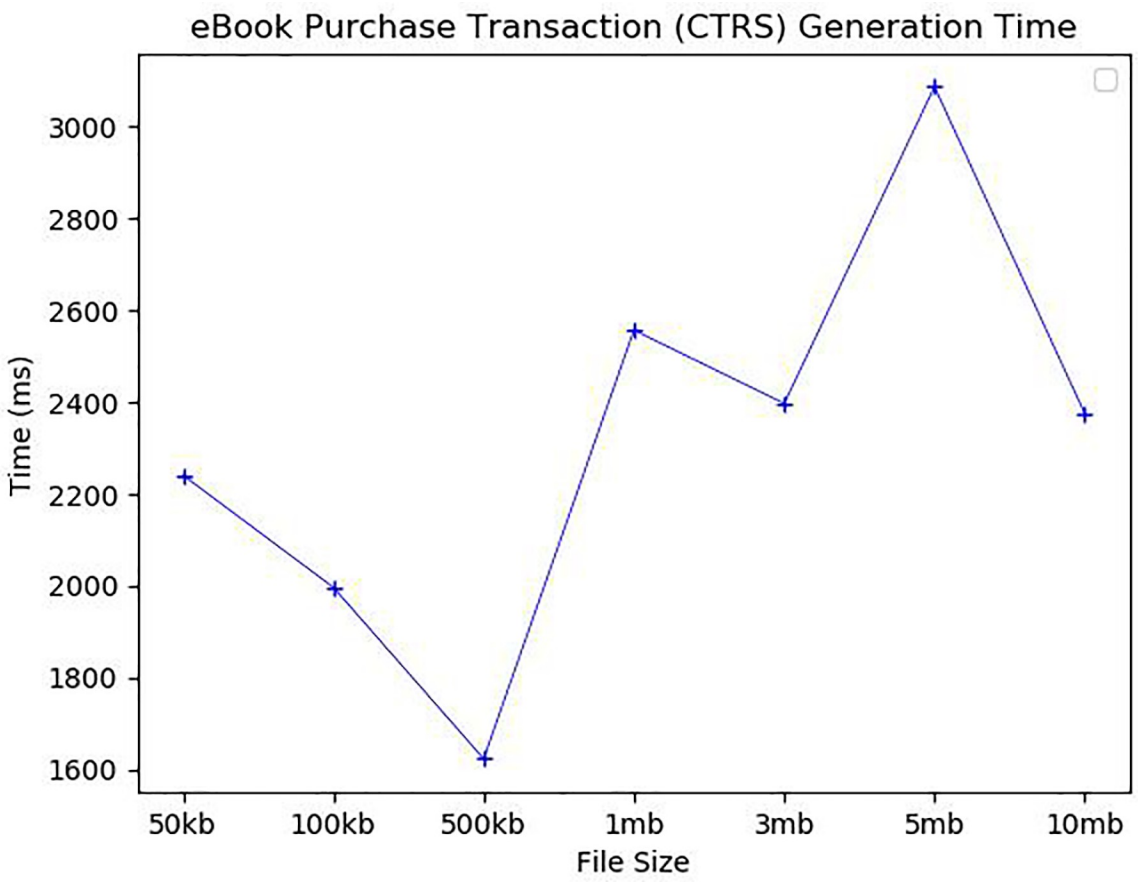

Fig 14. eBook purchase transaction (*CTRS*) generation time.

https://doi.org/10.1371/journal.pone.0228418.g014

and that block is then added to the existing blockchain. In the blockchain update, the mining block, which finds a random nonce satisfying a predefined difficulty, is the most time-consuming. We analyzed the blockchain update time according to levels of difficulty 3, 4, and 5. The block mining time was very jagged, but the average execution time was different depending on the level of difficulty as shown in Fig 15. For *B_Chain* update, it took about 37ms for level 3 of difficulty, 485ms for level 4, and 2862ms for level 5. Similarly, for *C_Chain*, it took about 47ms for level 3, 342ms for level 4, and 3340ms for level 5.

Fig 16 and Fig 17 show the average eBook download request message verification time and eBook download and resave time according to various eBook sizes. The eBook download also requires the verification of the eBook download request. We assumed that the book repository only keeps the hash values of all *CTRS*s recorded in *C_Chain*. Thus, the eBook download request message contains the corresponding *CTRS*. As shown in Fig 16, the *CTRS* validation took about 4.5ms regardless of eBook size. On the other hand, eBook download time is mainly affected by the number of eBook pieces and the size of each file piece, so it increases significantly with the number of eBook pieces. It took about 57ms and 64ms for 50 kb and 100 kb eBooks, respectively, but it took 110ms for 500 kb, 250ms for 1 mb, 374ms for 3 mb, 797ms for 5 mb, and 1473ms for 10 mb. The downloads took less time than the uploads because the download only requires communication and not include the file storage step. The downloaded encrypted book pieces are re-encrypted with a randomly chosen new book key and saved in the user's local storage as encrypted. More precisely, each book piece must be decrypted and

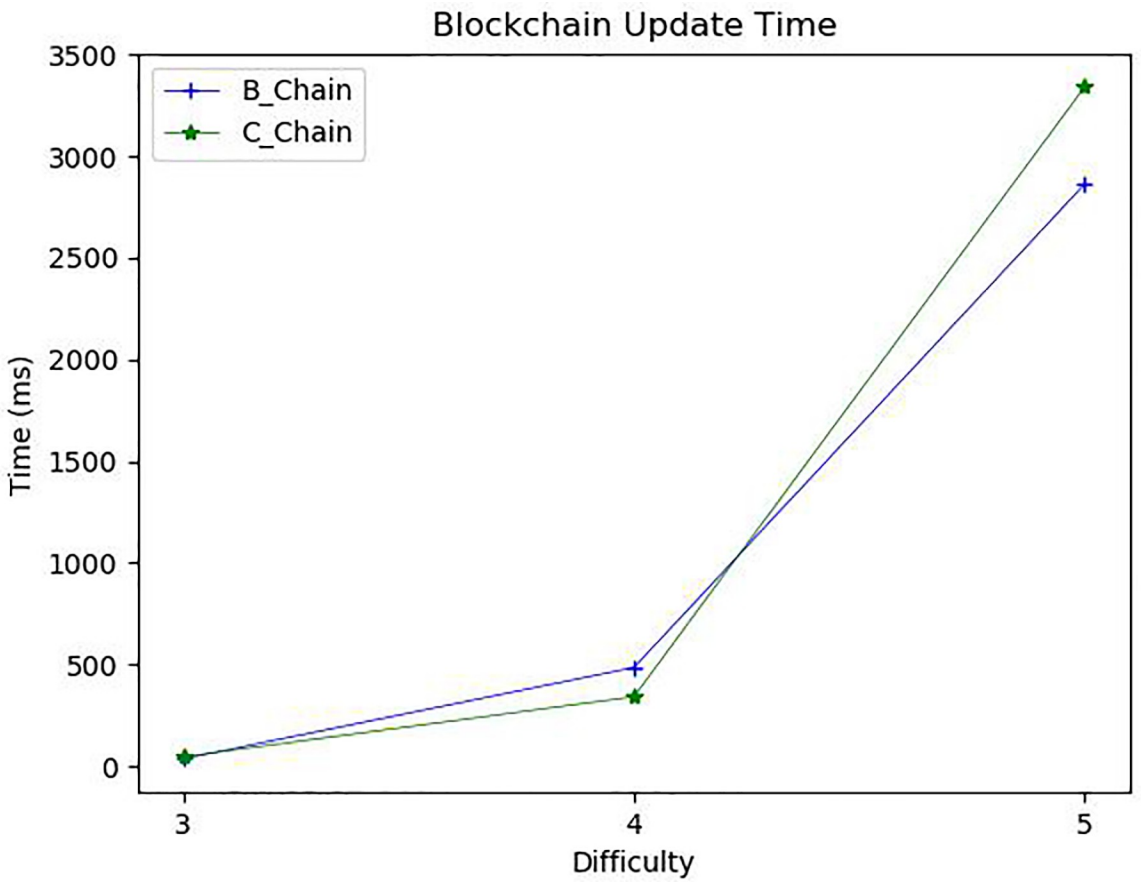

Fig 15. Blockchain update time.

https://doi.org/10.1371/journal.pone.0228418.g015

then combined into whole content, a new book key should be generated, the whole content must be encrypted again with the new book key, and finally, both the re-encrypted book content and its book key are saved in files. Thus, this task increases in proportion to the number of file pieces and the size of each file piece, and in fact, it took about 20ms for a 50 kb eBook, 27ms for 100 kb, 114ms for 500 kb, 352ms for 1 mb, 1145ms for 3 mb, 2990ms for 5 mb, and 11386ms for 10 mb. Our simulated results show that every action related to direct eBook transaction can be accomplished in a reasonable time.

Finally, we compare the features of the proposed model with Publica [37], which is the most similar blockchain-based peer-to-peer eBook publishing product. Table 8 summarizes the features of both models.

The proposed model is meaningful in that it protects the copyright of paid contents as well as the safety of direct payment based on blockchain technology in trustless environment, and provides the function that buyers can directly verify the validity of paid contents and the corresponding book keys during purchasing.

Conclusions

We have here introduced a blockchain-based eBook transaction system that enables direct eBook purchase between authors and readers without trusted parties. In the proposed system, any user can be either an author or an eBook content buyer. Because of the direct transaction,

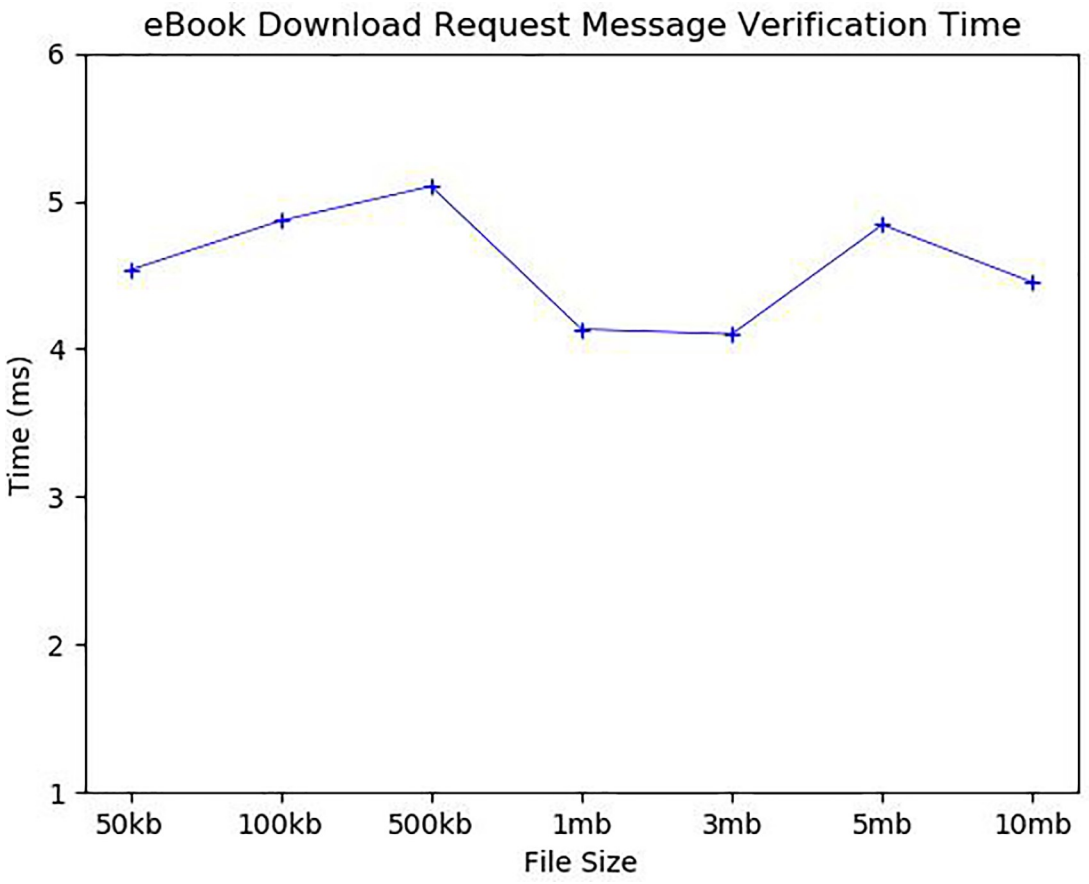

Fig 16. eBook download request message verification time.

https://doi.org/10.1371/journal.pone.0228418.g016

no intermediate costs occur, but the security of eBook contents and eBook purchase transactions cannot be protected. Thus, we proposed a secure and reliable eBook direct transaction system that overcomes all security vulnerabilities in the absence of trust parties.

The proposed system guarantees the confidentiality of eBook contents; only a legitimate purchaser can read a purchased book. Because the buyer can verify the validity of the eBook content before the purchasing process is completed, authors are also prevented from publishing garbage content. Forging and repudiating previously performed transactions are prevented by the blockchain technology, making eBook purchase transactions securely protected. Because every purchased eBook's content is newly encrypted and stored by the buyer's service application, resale or redistribution of the purchased content is systemically prevented. As such, the proposed system secures not only the contents of eBooks but also the rights of buyers and authors.

We examined the security of the proposed protocols and the average execution time for main actions including eBook registration, purchase, downloading, and storage, according to various sizes of eBook contents and different levels of difficulty in blockchain generation. Our implemented results show that all the main operations could be accomplished in a reasonable time.

Currently, only eBook purchases are allowed, but continuous research will be conducted to expand the system to allow eBook rental for a limited time. In addition, studies for evaluating the reputation of eBooks and authors will continue as well.

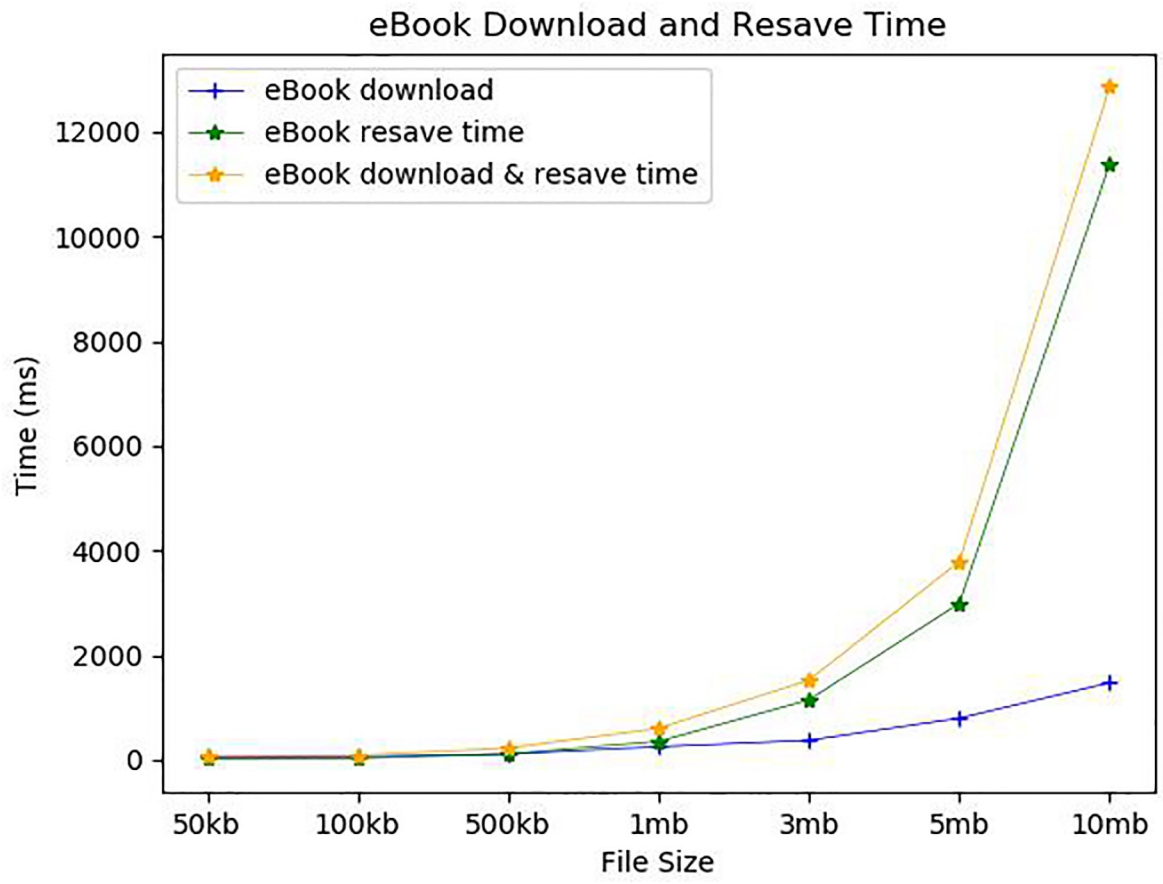

Fig 17. eBook download and resave time.

https://doi.org/10.1371/journal.pone.0228418.g017

Table 8. Summary of the features of the proposed model and Publica.

Features	Publica [37]	The proposed model
Trustee	Publica	X
Self-publishing	O	O
Direct payment to author	O	O
Payment	Digital coin	Digital coin
Copyright management	Publica, Publisher or Author	Blockchain
Purchase contract management	Blockchain	Blockchain
Right to read	Book token	Book key
Book key verification during purchase	X	O
Content verification during purchase	X	O
Content re-encryption	unknown	O

https://doi.org/10.1371/journal.pone.0228418.t008

Author Contributions

Conceptualization: Soyoung Park.

Methodology: Soyoung Park.

Project administration: Soyoung Park.

Software: Jeonghee Chi, Jangyeon Lee, Nakyung Kim, Jeewoo Choi, Soyoung Park.

Writing – original draft: Soyoung Park.

Writing – review & editing: Soyoung Park.

References

1. Watson A. E-books–Statistics & Facts, Dec 18, 2018, statista.com, Available from: https://www.statista.com/topics/1474/e-books/
2. RIDI Corp. RIDIBOOKS [Internet]. Available from: https://ridibooks.com/
3. Millie Corp. Millie [Internet]. Available from: https://www.millie.co.kr
4. NAVER WEBTOON Corp. SERIES [Internet]. Available from: https://series.naver.com/series
5. Watson A. Number of self-published books in the United States from 2008 to 2017, by format, Nov 20, 2018, statista.com, Available from: https://www.statista.com/statistics/249036/number-of-self-published-books-in-the-us-by-format/
6. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008.
7. The Bitcoin Project [Internet]. Available from: https://bitcoin.org/
8. The Ethereum Project [Internet]. Available from: https://www.ethereum.org
9. The Hyperledger Project [Internet]. Available from: https://www.hyperledger.org/
10. Tschorsch F, Scheuermann B. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. IEEE Communications Surveys & Tutorials. 2016; 18(3): 2084–2123. https://doi.org/10.1109/COMST.2016.2535718
11. O'Hara K. Smart Contracts—Dumb Idea. IEEE Internet Computing. 2017; 21: 97–101. https://doi.org/10.1109/MIC.2017.48
12. Kosba A, Miller A, Shi E, Wen Z, Papamanthou C. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. IEEE Symposium on Security and Privacy (SP), 2016; 839–858. https://doi.org/10.1109/SP.2016.55
13. Watanabe H, Fujimura S, Nakadaira A, Miyazaki Y, Akutsu A, Kishigami J. Blockchain contract: Securing a blockchain applied to smart contracts. 2016 IEEE International Conference on Consumer Electronics (ICCE). 2016; 7–11. https://doi.org/10.1109/ICCE.2016.7430693
14. Crosby M, Nachiappan, Pattanayak P, Verma S, Kalyanaraman V. Blockchain Technology: Beyond Bitcoin. Applied Innovation Review. 2016; 2: 6–19
15. Zyskind G, Nathan O, Pentland A. Decentralizing Privacy: Using Blockchain to Protect Personal Data, IEEE Security and Privacy Workshops. 2015; 180–184. https://doi.org/10.1109/SPW.2015.27
16. Kosba A, Miller A, Shi E, Wen Z, Papamanthou C. Hawk: The blockchain model of cryptography and privacy-preserving smart contract. 2016 IEEE Symposium on Security and Privacy. 2016. https://doi.org/10.1109/SP.2016.55
17. Wang S, Zhang Y, Zhang Y. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. IEEE Access. 2018; 6: 38437–38450. https://doi.org/10.1109/ACCESS.2018.2851611
18. Ma Z, Jiang M, Gao H, Wang Z. Blockchain for digital rights management. Future Generation Computer Systems. 2018; 89: 746–764. https://doi.org/10.1016/j.future.2018.07.029
19. Koblitz N. Elliptic Curve Crytosystems. Mathematics of Computation. 1987; 48: 203–209.
20. Miller VS. Use of Elliptic Curves in Cryptography. Advances in Cryptology—CRYTO '85. 1985; 128: 417–426.
21. IEEE Standard Specifications for Public Key Cryptography, IEEE Std 1363–2000. The Institute of Electrical and Electronics Engineers, Inc. 2000. Available from: https://perso.telecom-paristech.fr/guilley/recherche/cryptoprocesseurs/ieee/00891000.pdf
22. Li J. On peer-to-peer content delivery. Peer-to-Peer Networking and Applications. 2008; 1(1): 45–63. https://doi.org/10.1007/s12083-007-0003-1
23. Hartung F, Kutter M. Multimedia watermarking techniques. Proceedings of the IEEE. 1999; 87 (7): 1079–1107
24. Cox IJ, Miller M, Bloom J, Fridrich J. Kalker T. Digital watermarking and steganography (2nd Ed). Morgan-Kaufmann. 2008.
25. Bianchi T, Piva A. Secure watermarking for multimedia content protection: A review of its benefits and open issues. IEEE Signal Processing Megazine. 2013; 30 (2): 87–96

26. Barg A, Blackley GR, Kabatiansky GA. A digital fingerprinting codes: problem, statements, constructions, identification of traitors. IEEE Transactions on Information Theory. 2003; 49: 852–865. https://doi.org/10.1109/TIT.2003.809570

27. Voloshynovskiy S, Farhadzadef F, Koval O, Holotyak T. Active Content fingerprinting: A marriage of digital watermarking and content finger printing. 2012 IEEE International Workshop on Information Forensics and Security (WIFS). 2012; 175–180. https://doi.org/10.1109/WIFS.2012.6412645

28. Li J, Hsieh C, Hung C. A novel DRM framework for peer-to-peer music content delivery. The Journal of Systems and Software. 2010; 83: 1689–1700. https://doi.org/10.1016/j.jss.2010.04.071

29. Qureshi A, Megias D, Rifa-Pous H. Framework for preserving security and privacy in peer-to-peer content distribution systems. Expert Systems with Applications. 2015; 42: 1391–1408

30. Deng M, Preneel B. On secure and anonymous buyer-seller watermarking protocol. 2008 Third International Conference on Internet and Web Applications and Services. 2008; 524–529. https://doi.org/10.1109/ICIW.2008.28

31. Rial A, Deng M, Bianchi T, Piva A, Preneel B. A provably secure anonymous buyer-seller watermarking protocol. IEEE Transactions on Information Forensics and Security. 2010; 5(4): 920–931. https://doi.org/10.1109/TIFS.2010.2072830

32. Phan RCW, Goi BM, Poh GS, Kim J. Analysis of a buyer-seller watermarking protocol for trustworthy purchasing of digital contents. Wireless Personal Communications. 2011; 56(10): 73–83. https://doi.org/10.1007/s11277-009-9876-z

33. Tsudik G. Message authentication with one-way hash functions. ACM Computer Communication Review. 1992; 22(5): 29–38. https://doi.org/10.1145/141809.141812

34. Deswarte Y, Quisquater JJ, Saidance A. Remote integrity checking. Integrity and Internal Control in Information Systems VI. IICIS 2003. 2004; 140. https://doi.org/10.1007/1-4020-7901-X_1

35. Abbdal SH, Jin H, Zou D, Yassen AA. Secure third party auditor for ensuring data integrity in cloud storage. 2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops. 2014; https://doi.org/10.1109/UIC-ATC-ScalCom.2014.17

36. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, et al. Provable data possession at untrusted stores. Proceedings of the 14th ACM conference on Computer and Communications Security (CCS07). 2007; 598–609. https://doi.org/10.1145/1315245.1315318

37. Publica [Internet]. Available from: https://publica.com/

38. SHA256 Standard. Federal Information Processing Standards (FIPS) PUB 180–2. National Institute of Standards and Technology (NIST). 2002. Available from: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

39. Courtois NT, Grajek M, Naik R. Optimizing SHA256 in Bitcoin Mining. Cryptography and Security Systems (CSS 2014). 2014; 448: 131–144. https://doi.org/10.1007/978-3-662-44893-9_12

40. Daemen J, Rijmen V. The Design of Rijndael: AES—The Advanced Encryption Standard. Springer Science & Business Media, 2002.

41. Announcing the ADVANCED ENCRYPTION STANDARD (AES). Federal Information Processing Standards (FIPS) PUB 197. National Institute of Standards and Technology (NIST). November 26, 2001. Retrieved October 2, 2012. Available from: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf