Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

RCDS: a right-confirmable data-sharing model based on symbol mapping coding and blockchain*

Liang WANG†1, Shunjiu HUANG†‡1, Lina ZUO[1], Jun LI[2], Wenyuan LIU[3]

[1]*School of Cyber Security and Computer, Hebei University, Baoding 071000, China*

[2]*Xiong'an Intelligent City Innovation Federation, Xiong'an 071700, China*

[3]*School of Information Science and Engineering, Yanshan University, Qinhuangdao 066000, China*

†E-mail: wangl@hbu.edu.cn; sjhuang1120@stumail.hbu.edu.cn

Received Dec. 24, 2022; Revision accepted Mar. 27, 2023; Crosschecked Aug. 10, 2023

Abstract: The problem of data right confirmation is a long-term bottleneck in data sharing. Existing methods for confirming data rights lack credibility owing to poor supervision, and work only with specific data types because of their technical limitations. The emergence of blockchain is followed by some new data-sharing models that may provide improved data security. However, few of these models perform well enough in confirming data rights because the data access could not be fully under the control of the blockchain facility. In view of this, we propose a right-confirmable data-sharing model named RCDS that features symbol mapping coding (SMC) and blockchain. With SMC, each party encodes its digital identity into the byte sequence of the shared data by generating a unique symbol mapping table, whereby declaration of data rights can be content-independent for any type and any volume of data. With blockchain, all data-sharing participants jointly supervise the delivery and the access to shared data, so that granting of data rights can be openly verified. The evaluation results show that RCDS is effective and practical in data-sharing applications that are conscientious about data right confirmation.

Key words: Data right confirmation; Symbol mapping coding; Blockchain; Data sharing; Traitor tracing; Access control

https://doi.org/10.1631/FITEE.2200659 **CLC number:** TP309.2

1 Introduction

1.1 Background

The growth of the digital economy relies on trusted data sharing in which data right confirmation (DRC) remains a challenge. Data sharing is not equivalent to data ownership transferring. The ownership and use rights of the shared data should be correctly confirmed; otherwise, no one will be willing to share information with others. Unfortunately, transaction repudiation (Zhang R et al., 2023) and data piracy (Barni and Bartolini, 2004) are still the worst adversaries of DRC. To withstand them, we need to put into effect a dependable DRC scheme that integrates more credible methods of traitor tracing (Zhang LY et al., 2020) and access control.

However, the shortcomings of existing DRC approaches in traditional data-sharing models cannot be ignored. First, most of those models depend on trusted third parties (TTPs) (Coffey and Saidha, 1996), which may not be that trustworthy. Then, some models are built on staged encryption (Ali et al., 2016), which is usually inefficient when dealing with a large volume data. Typically, digital watermarking based DRC schemes work only for sharing certain types of data (Wang HL et al., 2018), which

‡ Corresponding author

* Project supported by the Natural Science Foundation of Hebei Province, China (No. F2023201032) and the S&T Program of Hebei Province, China (No. 20310105D)

ORCID: Liang WANG, https://orcid.org/0000-0003-2839-0832; Shunjiu HUANG, https://orcid.org/0000-0003-1121-1842

could be detrimental to the availability of the ubiquitous undistorted data. The absence of watermark forgery supervision mechanisms is also a major limitation of those schemes.

Recently, researchers proposed using blockchain as a distributed TTP to record data-sharing processes, aiming at providing DRC services (Zha et al., 2020; Zhao et al., 2021). Those blockchain-based data-sharing models support DRC to some extent, but they have limited control over the data-sharing processes. Moreover, those models either store excessive data on blockchain or require complicated encryption computation, and thus often result in high costs and low benefits in practice.

Considering the above deficiencies, we propose a new data-sharing model, RCDS, which combines symbol mapping coding (SMC) and blockchain. SMC is a method that encodes data holders' fingerprints into the byte sequence of the data copy. It allows data right granting to be independent of data content, so RCDS can work on any type of data. Blockchain in RCDS distinctively records the key mapping elements generated by SMC, and offers reliable evidence when confirming data rights. In RCDS, the blockchain witnesses the whole process of data sharing, and endows traitor tracing and access control with provable credibility.

1.2 Related works

Existing DRC schemes for data sharing can be divided into three groups: TTP-based schemes, phased encryption schemes, and blockchain-based schemes. We review them briefly below.

1.2.1 TTP-based schemes

TTP-based schemes usually employ third parties to supervise the communication among data-sharing participants and to offer DRC proofs (Zhu ZM and Jiang, 2016; Frattolillo, 2017; Ganesh et al., 2017). TTPs in these schemes play the role of evidence verifier. Once disputes occur, a TTP will provide testimony for arbitration. The problem is that the so-called TTP may not be that trusted. Coffey and Saidha (1996) first proposed a TTP-based scheme for the general non-repudiation problem. In this scheme, the third party was relied upon without reserve, which may increase the possibility of collusion attacks. To solve this problem, Zhu ZM and Jiang (2016) introduced an anti-collusion attack data-sharing model based on an asymmetric cryptosystem and the Delov–Yao model. However, this model could not resist man-in-the-middle attack or data-tampering attacks because servers did not verify user registration (Ganesh et al., 2017). Moreover, the single point of failure is a problem that cannot be ignored under the centralized architecture. In summary, TTP-based schemes cannot meet the requirements of DRC because data can easily be tampered with and third parties are not fully trusted.

1.2.2 Phased encryption schemes

A phased encryption scheme implements reliable data sharing usually by encrypting shared data in phases (Ali et al., 2016; Zaghloul et al., 2020). In this scheme, shared data are divided into several parts and delivered piece by piece. Once a piece is received, an ACK must be returned from the receiver to the sender before the next piece can be sent. Then, the ACK is used as non-repudiation evidence. The drawback of this scheme is that too many rounds of communication could be needed in one data sharing instance. Ali et al. (2016) proposed a two-stage non-repudiation protocol, but the problem of low efficiency remained. Furthermore, TTPs assumed by these schemes are unrealistic.

1.2.3 Blockchain-based schemes

Blockchain has been introduced into data sharing as an infrastructure to provide irrefutable evidence of DRC (Huckle and White, 2017; Gong et al., 2019; Zha et al., 2020). Blockchain's excellent features are often used to produce traceability of data sources and their sharing histories (Saini et al., 2021; Zhao et al., 2021). Blockchain and digital watermarking are combined to improve the security of copyright protection and data source tracing (Wang HL et al., 2018; Qian et al., 2019). Wang HL et al. (2018) proposed a combinatorial model, using first a data holding proof method to audit data integrity and then a digital watermarking scheme to confirm the origin of the shared data. However, this degree of combination is still insufficient to fight against data piracy, because access control is not emphasized.

Some researchers combined blockchain with encryption systems to improve access control in data sharing (Ersoy et al., 2021; Sifah et al., 2021), but it

is usually not efficient when faced with large volumes of data. Wang S et al. (2022) proposed a big-data sharing scheme that uses smart contracts to execute access rules. However, encryption and decryption require a lot of time for a large amount of data, resulting in low efficiency.

In addition, zero knowledge proof (ZKP) and non-fungible token (NFT) inspired researchers to develop some featured DRC methods. Some new studies constructed ZKP models to prove the ownership or use right of assets by third parties (Cao and Zhao, 2021; Sun et al., 2021; Lin et al., 2022). Some other studies generated mainly unique digital certificates (i.e., NFTs) to claim ownership of specific data assets and achieved secure data sharing by selling those NFTs. However, the above techniques have some intolerable shortcomings in DRC data sharing. For example, ZKP construction often has low efficiency and poor scalability (Giacomelli et al., 2016; Parno et al., 2016), and NFT is still a technically immature concept (Okonkwo, 2021).

1.3 Contributions

The contributions of our work to research on DRC in the process of data sharing are as follows:

1. We propose SMC to make fingerprint encoding suited to any type of data content. It prevents fingerprint forgery and enhances the credibility of DRC in the process of data sharing.

2. RCDS enables data-sharing processes to be supervised on blockchain publicly by empowering trusted traitor tracing and access control.

2 Preliminaries

2.1 Basic concepts

Here, we first introduce relevant technical concepts that will be used in the RCDS model.

1. Transaction repudiation. There are currently two types of transaction repudiation (Chen et al., 2022; Wang L et al., 2022) in the process of data sharing: repudiation of sending (RoS) and repudiation of receipt (RoR). RoS occurs in a situation like this: Alice (the data sender) fabricates a record in which she sent data to Bob (the data receiver), thereby imposing responsibility for data security on Bob. RoR is another case: Bob denies the fact that he received data from Alice, thereby shirking his responsibility for data security. For non-repudiation, all participants must deny nothing about their behaviors.

2. Data piracy. Data piracy refers to the acts of reproducing and redistributing data copies without the consent or authorization of data owners. It is tricky because of the replicability of the data. Anti-piracy requires some control over access to data.

3. Traitor tracing. Traitor tracing is a common countermeasure against transaction repudiation. It does not prevent users from denying what they have done, but tracks and obtains evidence of what they have done. It is often a strategy of pre-deterrence and post-accountability rather than prevention, and DRC is the underlying logic of this strategy.

4. Access control. Access control refers to policies that prevent unauthorized access to data. Authentication and fine-grained data encryption are common methods for access control, and are generally premised on DRC.

2.2 Blockchain

Blockchain technology is capable of ensuring the security of data transmission and access by multi-party co-maintenance and cryptography, especially consistency achievement of data storage, tamper-resistance of records, and anti-repudiation of data delivery (Zhu LH et al., 2019). Therefore, blockchain systems are expected to facilitate data sharing in a more credible way than traditional cloud- and exchange-based systems (Gai et al., 2018). Generally, multi-party co-maintenance and redundant storage provide blockchain systems with decentralization. At the same time, tamper-resistance and anti-repudiation embody the reliability of data sharing.

2.3 Symbol mapping coding

SMC is the innovative underpinning of our work. For ease of understanding, Table 1 lists the notations used in this paper.

Given a data object, SMC works by dividing the byte sequence of the data object into symbols and recoding them using a generated symbol mapping table (SMT). SMT maps each symbol into two different types of digital codes: one is called plain code (which is used to encode symbols) and the other is called hidden code (which is used to carry fingerprints). Unlike ordinary encryption, SMC uses SMT instead

Table 1 Notations used in this paper

Notation	Connotation
SMT	Symbol mapping table
SMT_X	SMT of user X
SMT_X^+	Private part of SMT_X
SMT_X^-	Public part of SMT_X
SK_X	Private key of user X
F_X	Fingerprint of user X
d	Data object
d^*	Partial data acquired through a query
d^F	Data object d with fingerprint F embedded
$\otimes$	Encapsulated decoder
DDes	Data description
txn	Transaction on blockchain
$h(\cdot)$	Hash function
$\lvert\cdot\rvert$	Length of a set
θ	Symbol length measured in bytes per symbol
ρ	Plain code length measured in bytes per code
η	Hidden code length measured in bytes per code
φ	Fingerprint redundancy indicating the strength of fingerprint embedding
γ	Redundancy factor measured in fingerprints per symbol
λ	Symbol length factor restricting the co-domain of θ
χ	Coding redundancy for obfuscating symbol maps
τ	Upper limit of the storage consumption

Table 2 Schematic symbol mapping table

Symbol	Plain code	Hidden code
A(0x41)	0x0041	W(0x57)
	0xE410	O(0x4F)
E(0x45)	0xE451	M(0x4D)
	0xE452	R(0x52)
I(0x49)	0x004C	*(0x2A)
	0xE4C1	L(0x4C)
O(0x4F)	0xE4F0	D(0x44)
	0x004F	$(0x24)
U(0x55)	0x004D	!(0x21)
	0x0056	P(0x50)

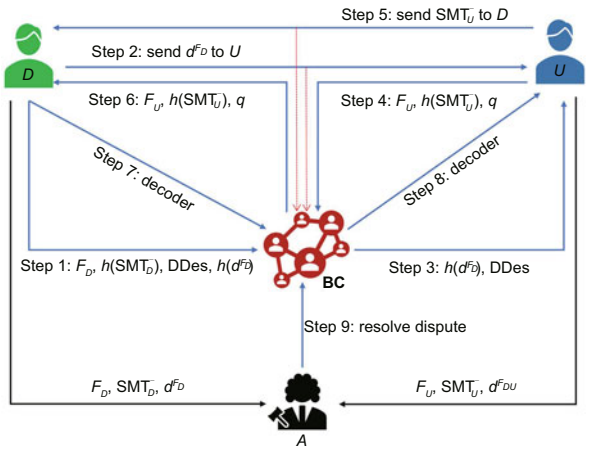

Fig. 1 Overview of RCDS

D: distributor; U: authorized user; A: arbiter; BC: blockchain; d: data object; d^F: data object with fingerprint F embedded; F_X: fingerprint of user X; SMT_X: symbol mapping table (SMT) of X; SMT_X^-: public part of SMT_X; q: data query; d^*: output of q through encapsulated decoder $\otimes$

of a secret key. To make SMT difficult to guess, we generate it using one-way mapping.

Specifically, an SMT includes a symbol set, a plain code set, and a hidden code set. Each symbol will be linked to at least one plain code, so will each hidden code. Let S be the symbol set, P the plain code set, and H the hidden code set. An SMT should meet the following two irreversible surjections:

1. $\delta_1 : p \to s,\ p \in P,\ s \in S$;
2. $\delta_2 : p \to h,\ p \in P,\ h \in H$.

Table 2 is an explanatory SMT. Given the string AAEIOU, the code "0x0041 0xE410 0xE452 0xE4C1 0xE4F0 0x004D" mapped with Table 2 carries the hidden string WORLD!. In this way, we can generate an SMT for any given data content to hide a string that is used as a fingerprint.

3 RCDS model

RCDS is a data-sharing model that achieves reliable DRC by identifying fingerprints in data copies. Fig. 1 shows the working principle of this model, and we describe its workflow in detail below.

1. Distributor D generates an SMT_D for data object d with his/her fingerprint F_D and private key hash $h(\mathrm{SK}_D)$, encodes d into d^{F_D} with the generated SMT_D, and records $h(d^{F_D})$, F_D, $h(\mathrm{SMT}_D^-)$, and DDes onto blockchain BC.

2. D sends d^{F_D} to user U through the off-chain channel, and uploads the transaction record to BC.

3. Authorized user U obtains $h(d^{F_D})$ and DDes from BC.

4. U generates an SMT_U for data object d^{F_D} with his/her fingerprint F_U and private key hash $h(\mathrm{SK}_U)$, encodes d^{F_D} into $d^{F_{DU}}$ with the generated SMT_U, redacts query q according to DDes, and records F_U, $h(\mathrm{SMT}_U^-)$, and q onto BC.

5. U sends SMT_U^- to D through the off-chain channel, and uploads the transaction record to BC.

6. D obtains $h(\mathrm{SMT}_U^-)$, F_U, and q from BC.

7. D encapsulates the access control policy into a decoder and uploads the decoder to BC.

8. U obtains the decoder from BC.

9. The arbiter judges the repudiation behavior with the interactive records on the blockchain and fingerprint verification results.

In this model, a data delivery process between D and U can be formalized as the following steps:

1. $\mathrm{SMT}_D \leftarrow \phi_1\left(d, h\left(\mathrm{SK}_D\right), F_D, \gamma, \tau, \lambda\right)$;
2. $d^{F_D} \leftarrow \phi_2\left(d, \mathrm{SMT}_D, F_D\right)$;
3. $\mathrm{SMT}_U \leftarrow \phi_1(d^{F_D}, h\left(\mathrm{SK}_U\right), F_U, \gamma, \tau, \lambda)$;
4. $d^{F_{DU}} \leftarrow \phi_2(d^{F_D}, \mathrm{SMT}_U, F_U)$.

Obviously, neither d^{F_D} nor $d^{F_{DU}}$ is directly readable without correct SMTs. The authorized user can access data object d by the following formalized steps:

1. for D:
$$\mathrm{SMT}^+_{D^*} \leftarrow \phi_3\left(\mathrm{SMT}^+_D, q\right),$$
$$\otimes \leftarrow \phi_4\left(\mathrm{SMT}^+_{D^*}, F_U, \mathrm{SMT}^-_U\right);$$
2. for U:
$$d^* \leftarrow \phi_5\left(d^{F_{DU}}, \mathrm{SMT}^+_U, \otimes\right).$$

When an authorized user wants to access d, he/she should generate a query statement q according to the data description DDes and apply for a decoder $\otimes$ from the distributor. The user then obtains the data he/she needs according to $\otimes$, rather than directly obtaining the whole data from $d^{F_{DU}}$. Therefore, we design ϕ_4 which encapsulates the access control policy and publishes only minimal query interfaces that do not disclose the original data. We also design ϕ_5 to ensure that only the correct user can query information from the data.

The arbiter can use the tester on $d^{F_{DU}}$ and d^{F_D} to identify whether the holder of d is an authorized user or a distributor:

1. $\mathrm{Tester}_U = \phi_6\left(d^{F_{DU}}, \mathrm{SMT}^-_U, F_U, \varepsilon, \varsigma, \tau, r\right)$;
2. $\mathrm{Tester}_D = \phi_6\left(d^{F_D}, \mathrm{SMT}^-_D, F_D, \varepsilon, \varsigma, \tau, r\right)$.

Each of the above testers returns either negative or positive. Positive means that a fingerprint is detected from $d^{F_{DU}}$ or d^{F_D}, and negative means the opposite. From Fig. 1, it is easy to understand that each party of data delivery holds a unique copy of the data, in which the party's own fingerprint is embedded and the user's copy of data is confidential to the distributor. ϕ_6 is the fingerprint identification algorithm that is the core of DRC. Any node in the blockchain can act as an arbiter to identify the fingerprints from d to check users' permissions.

In this model, we use the blockchain instead of a TTP, and use it to provide a complete evidence chain for each data-sharing process. Nodes of the blockchain network are exactly participants in data-sharing activities. Through consensus, they jointly maintain the consistency of the blockchain. The blockchain immutably records all key elements of each data delivery transaction to make the model credible.

4 Model construction

In RCDS, the sharing of a data object involves a series of processes including SMT generation, fingerprint embedding, fingerprint identification, and data query. Based on implementing these processes, the following subsections present the construction of RCDS.

4.1 SMT generation

Unlike watermarking on images, fingerprints are embedded with the help of an SMT, which works to encode and decode the raw data like a codebook. Fingerprints are encoded in the form of hidden codes along with the generation of SMTs. In turn, these fingerprints can be identified by parsing the encoded data with these SMTs. An SMT must be associated, one to one, with the corresponding data, so fingerprint identification can be unambiguous. With this in mind, we design ϕ_1 as described in Algorithm 1, where an SMT is divided into two parts: the private part, SMT^+, consisting of $\langle s_\theta, p_\rho\rangle$, and the public part, SMT^-, consisting of $\langle p_\rho, h_\eta\rangle$.

In Algorithm 1, the value of θ should be a random number and meet two conditions. The first condition is $\mathrm{SMT}^- \leq \tau$, where τ is an upper limit artificially designed for SMT^- in practice. The second condition is $|d^F| \geq \lambda|F|$, ensuring that there is enough encoding space of d^F at the side of U. To ensure enough encoding space, we assume that the explicit code space is twice the symbol space. According to the given τ and λ, we can set the range of θ to satisfy the following conditions:

$$\begin{cases} 2|S| \leq 2^{8\rho}, \\ |S| = \dfrac{|d|}{\theta}, \\ \rho|S| = |d^F|, \\ |d^F| \geq \lambda|F|, \\ \mathrm{SMT}^- \leq \tau. \end{cases}$$

The function symbolize$(\cdot)$ is used to randomize the value of θ within the range defined by

Algorithm 1 Symbol mapping table generation (ϕ_1)

Input: d, $h(\mathrm{SK})$, F, γ, τ, λ
Output: SMT
1: $\theta \leftarrow \mathrm{symbolize}\left(h\left(\mathrm{SK}\right), |F|, |d|, \tau, \lambda\right)$
2: $S \leftarrow \mathrm{Pt}_\theta d$
3: $\rho \leftarrow \lceil (\log_2 |S| + 1)/8 \rceil$
4: $\eta \leftarrow \mathrm{customize}\left(h\left(\mathrm{SK}\right), |S|, |F|, \gamma\right)$
5: $\varphi \leftarrow \lceil \eta |S|/|F| \rceil$
6: $F_\eta \leftarrow \mathrm{Pt}_\eta F^\varphi$, where $F^\varphi \leftarrow \bigcup_\varphi F$
7: **for** each $s \in S$ and $h \in F_\eta$ **do**
8: generate a plain code $p_\rho \notin T$
/* T is a de-duplication set */
9: **if** $\mathrm{SMT}|_{s,h} = \varnothing$ **then**
10: add $\langle s, p_\rho, h \rangle$ to SMT
11: add p_ρ to T
12: **end if**
13: **end for**
14: $\chi \leftarrow \mathrm{obfuscate}\left(h\left(\mathrm{SK}\right), |\mathrm{SMT}^-|, \rho\right)$
15: **for** $i = 1$ to χ **do**
16: $s \leftarrow \mathrm{rand}\left(S\right)$
17: $h \leftarrow \mathrm{rand}\left(F_\eta\right)$
18: generate a plain code $p_\rho \notin T$
19: add $\langle s, p_\rho, h \rangle$ to SMT
20: add p_ρ to T
21: **end for**

inequality (1):

$$\left\lceil \frac{|d|}{\tau} \left(|F| + \frac{\log_2 |d| + 1}{8} \right) \right\rceil \le \theta \le \left\lfloor \frac{\rho |d|}{\lambda |F|} \right\rfloor. \quad (1)$$

Given $\gamma \in (0, 1]$, the fingerprint redundancy φ should be evaluated in the following range to obtain sufficient strength of fingerprint embedding:

$$\lceil \gamma |S| \rceil \le \varphi \le |S|. \quad (2)$$

Inequality (2) conforms to the constraint of being over perfect; i.e., if $\varphi > |S|$ were true, each symbol of the data object would match more than one fingerprint, which would add to the size of SMT.

It is clear that $\varphi |F| = \eta |S|$, so the range of η can be calculated with inequality (3):

$$\frac{|F|}{|S|} \lceil \gamma |S| \rceil \le \eta \le |F|. \quad (3)$$

Within this range, the function customize($\cdot$) is used to obtain a random value for η.

The function obfuscate($\cdot$) increases the redundancy of SMT to make θ harder to guess. The value of χ should not exceed the encoding space specified by ρ, so it can be calculated by

$$\chi \le \min\left(\left\lfloor 2^{8\rho} - |\mathrm{SMT}^-| \right\rfloor, \mathrm{c900} \right). \quad (4)$$

The value of c900 is derived from the third five-digit hexadecimal string in $h(\mathrm{SK})$. It is obtained by adding the first four characters together and multiplying the sum with the fifth string, so that the maximum result will not exceed 900.

The parameter ρ specifies the encoding space of the plain code and the expansion multiple of d. For example, when $\theta = 1$ and $\rho = 2$, 2 GB size of d^F will be obtained from 1 GB size of d. Therefore, ρ is usually calculated to meet the minimum requirement of the encoding space (line 3 in Algorithm 1).

Specifically, because the secrecy of θ should be maintained to make SMT difficult to forge, $h(\mathrm{SK})$ in Algorithm 1 makes the outputs of symbolize($\cdot$) (line 1 in Algorithm 1), customize($\cdot$) (line 4 in Algorithm 1), and obfuscate($\cdot$) (line 14 in Algorithm 1) user-dependent. This makes it more difficult to guess, because no users will reveal their private keys.

To facilitate authentication, we calculate the hash value of SMT^- and record $h(\mathrm{SMT}^-)$ on the blockchain, so that everyone in the network can obtain and verify SMT^- through $h(\mathrm{SMT}^-)$ and finally authenticate the corresponding data object through fingerprinting.

4.2 Fingerprint embedding

A data object should firmly carry its holder's fingerprint before it can be used. For a single delivery, the data holders include the distributor and the authorized user of the data copy. A fingerprint must be able to uniquely bind to a publicly verifiable identity. A reasonable way to make such a fingerprint is to generate it from the public key that can uniquely identify a data holder. Another factor that should not be overlooked is the size of the fingerprint. For a data object that is not very large, a fingerprint should not be too long, to avoid losing its robustness of embedding. Therefore, a viable way of fingerprint generation is to use the hash of the data holder's public key as a fingerprint and keep $|F| \le |d|$ true.

To enhance the strength of fingerprint embedding, it is better for a data object to be fully overlaid with redundant fingerprints. As inequality (2) requires, an η-partition (Pt_η) of the redundant fingerprints should be done to fit $|S|$ (line 6 in Algorithm 1).

In addition, the fingerprint of each user will be recorded on the blockchain and open to the network, so that everyone in the network can use these

fingerprints to check the identities of the shared data.

The embedding of redundant fingerprints is actually an encoding on data objects, and it is a two-step process. First, D pre-codes d through ϕ_2 to embed F_D. d^{F_D} is unable to be directly read and thus can be directly delivered to U. Then, U finally codes d^{F_D} through ϕ_2 to embed F_U. $d^{F_{DU}}$ allows U to query data through a dedicated decoder. Algorithm 2 details ϕ_2.

Algorithm 2 Fingerprint embedding (ϕ_2)

Input: d, SMT, F
Output: d^F
1: solve θ and η out of SMT
2: $S \leftarrow \mathrm{Pt}_\theta d$
3: $\varphi \leftarrow \lceil \eta |S|/|F| \rceil$
4: $F_\eta \leftarrow \mathrm{Pt}_\eta F^\varphi$, where $F^\varphi \leftarrow \bigcup_\varphi F$
5: **for** each $s \in S$ and $h \in F_\eta$ **do**
6: $p \leftarrow \mathrm{rand}(\{p | p \in \mathrm{SMT}|_{s,h}\})$
7: add p to d^F
8: **end for**

Algorithm 2 realizes a full coverage strategy in the encoding process. In the strategy, plain codes replace all the symbols of a data object, and hidden codes cover the duplicates of one fingerprint. Doing so can protect the original data from being leaked and make the cost of destroying a fingerprint unacceptable, because the data will not be recovered as usable if all the fingerprints hidden behind the data are broken. Another strategy in Algorithm 2 is randomized coding (line 6). It randomly chooses a plain code from SMT to match up with each hidden code, blocking the sequential guessing of the original symbols.

It is worth noting that the delivery of d^{F_D} is recorded as a main part of a transaction on the blockchain. Such records will be the compelling evidence for delineating suspects.

4.3 Fingerprint identification

The purpose of fingerprint identification is to test whether the data object contains the fingerprint of a specific user. Different from traditional digital watermarking, our method is based on hypothesis testing. In other words, for RCDS, it needs only to verify whether the data object contains specific fingerprints, and usually does not need to extract the fingerprints. We divide the fingerprint recognition algorithm into two parts: forward verification and backward verification. The details are as follows:

Algorithm 3 is a concrete solution to forward verification of ϕ_6. This solution collects the possible hidden code by processing the plain code part of the data, then divides the collected hidden code part into $|F|$ to obtain the possible fingerprint, and finally matches it with the user's fingerprint to obtain the match frequency. By calculating the Pearson correlation coefficient (Pan et al., 2021), at least one full match is needed to pass the verification.

Algorithm 3 Forward verification of ϕ_6

Input: d^F, SMT^-, F, ε, r
Output: negative/positive
1: solve ρ and η out of SMT^-
2: **if** $\left|d^F\right| > \tau\rho/\left(|F|+\rho\right)$ **then**
3: $\varepsilon \leftarrow \varepsilon\tau\rho/\left(\left(|F|+\rho\right)\left|d^F\right|\right)$
4: **end if**
5: $\varphi \leftarrow \min\left(\left\lfloor \eta\left|d^F\right|/(\rho|F|)\right\rfloor, \left\lfloor \eta\left|\mathrm{SMT}^-\right|/|F|\right\rfloor\right)$
6: **for** each $p_\rho \in d^F$ **do**
7: add $\mathrm{SMT}^-|_{p_\rho}$ to T
8: **end for**
9: $T_{|F|} \leftarrow \mathrm{Pt}_{|F|}T$
10: **for** each $F^* \in T_{|F|}$ **do**
11: $R_{F^*F} \leftarrow \mathrm{correlation}\left(F^*, F\right)$
12: **if** $|R_{F^*F}| \geq r$ **then**
13: $\varphi^* \leftarrow \varphi^* + 1$
14: **end if**
15: **end for**
16: **if** $\varphi^* \geq \varepsilon\varphi$ **then**
17: return positive
18: **else**
19: return negative
20: **end if**

When Algorithm 3 returns negative, it does not mean that there is no fingerprint match, because malicious processing on the encoded data might exist. So, we design Algorithm 4 to perform a backward verification, which estimates in reverse a set of possible plain codes of the complete redundant fingerprints, and compares the usage of each byte of these plain codes with that of the encoded data. Considering that a traitor may perform some special malicious operations on the data, such as adding 1 to each binary bit of the plain code, we analyze the difference between the bytes of the plain code and those of the malicious data to identify the embedded fingerprints.

The most complicated part of forward verification and backward verification is parameter optimization, which has a great impact on the effect

of fingerprint identification. The parameter ε is the lower limit ratio of fingerprint redundancy. It indicates that the minimum number of duplicate fingerprints should be detected correctly. However, the forward verification in Algorithm 3 is strict with the sequence of plain codes, so it is sensitive only to the correct d^F. When the forward verification of Algorithm 3 fails, the possibility of including fingerprints is not ruled out. So, at this time, reverse verification is another way to solve this situation. It uses the bytes of the complete string of redundant fingerprints as a function and observes to what extent the data to be processed conforms to this function. The parameter ς sets the upper limit of this range. Adjusting ε and ς will have a decisive impact on the fingerprinting. According to inequality (1), we can do the following derivation:

$$|d^F| = \rho|S| = \frac{\rho|d|}{\theta} \leq \frac{\tau\rho}{|F|+\rho}. \tag{5}$$

Obviously, if the above condition is not met, it is very likely that a deliberately manipulated piece of data is being managed. In Algorithms 3 and 4, parameters ε and ς are fine-tuned to increase the sensitivity of fingerprinting when the condition is not met.

4.4 Data query

For the data to be shared with RCDS, we use a secret encapsulation method to control the decoding of d^F. This method integrates data encryption and access control policies.

In Fig. 1, DDes is transmitted together with the encoded data. To enable users to obtain the data they need quickly and accurately, this description should be as detailed as possible, and at least include the parts shown in Fig. 2.

After an authorized user obtains d^{F_D} and DDes, they code their fingerprints into d^{F_D}. Then, they need to send a query statement q to the data distributor according to DDes to apply for access. To help distributors better encapsulate decoders, we design the following query primitives for the encapsulation:

First, the SELECT statement is the most frequently used and powerful query for relational databases. We use query primitives in structured query language (SQL) to query data for relational data described in DDes. See Fig. 3 for the specific query statement.

Algorithm 4 Backward verification of ϕ_6

Input: d^F, SMT^-, F, υ, ς
Output: negative/positive

solve ρ and η out of SMT^-
if $\left|d^F\right| > \tau\rho/\left(|F|+\rho\right)$ **then**
  $\varsigma \leftarrow \left(\varsigma\left(|F|+\rho\right)\left|d^F\right|\right)/(\tau\rho)$
end if
$U \leftarrow \left\{\langle b,u\rangle \middle| u \leftarrow \mathrm{count}\left(b\right), b \in d^F\right\}$
$F_\eta \leftarrow \mathrm{Pt}_\eta F^\varphi$, where $F^\varphi \leftarrow \bigcup_\varphi F$
for each $h \in F_\eta$ **do**
  $p \leftarrow \mathrm{rand}\left(\left\{p \middle| p \in \mathrm{SMT}^-|_h\right\}\right)$
  add p to d^{F^*}
end for
$\tilde{U} \leftarrow \left\{\langle b,u\rangle \middle| u \leftarrow \mathrm{count}\left(b\right), b \in d^{F^*}\right\}$
for each $\langle b,u\rangle \in \tilde{U}$ **do**
  $u' \in U|_b$
  add $|u-u'|$ to V
end for
$B \leftarrow \left\{\langle b,u\rangle \middle| u \leftarrow \mathrm{count}\left(b\right), b \in d^F\right\}$
$\tilde{B} \leftarrow \left\{\langle b,u\rangle \middle| u \leftarrow \mathrm{count}\left(b\right), b \in d^{F^*}\right\}$
$a \leftarrow B[1]/\tilde{B}[1]$ /* divided by the first number */
for each $\langle b,u\rangle \in \tilde{B}$ **do**
  $b' \in B|_b$
  if $b'/b = a$ **then**
    add 1 to Num
  end if
end for
if $\overline{V} \leq \varsigma$ and $\mathrm{Num}/|B| \geq \upsilon$ **then**
  return positive
else
  return negative
end if

Data type			Storage location	Data constraint	Data feature	Access policy
Relational data			Database	Integrity constraints	Attributes, dimensions, etc.	...
File data	Single media data	Text data	Data path	Text id	Text features	...
		Image data	Data path	Image id	Image features (color, attribute, etc.)	...
		Audio data	Data path	Audio id	Audio features (sound wave, etc.)	...
		Video data	Data path	Video structure (lens, etc.)	Video features (static features, etc.)	...
	Multimedia data		Content information (file type, entity semantion, implication relationship, etc.)	Feature information (attributes, entity similarity, etc.)	Spatial-temporal information (topological relationship, space calculation, etc.)	...

Fig. 2 DDes format

Second, to query file type data, we expand the query statements used in relational data by studying

the literature (Wu, 2009), so that it can accurately search in file type data. These extensions are manifested mainly in the WHERE clause. The WHERE clause describes the conditions that the target object should meet. The specific expression is shown in Fig. 4, in which the keywords are explained in Table 3.

For the query statement given above, we will further illustrate the language description ability of the query statement through a query example (mainly for file type data, because relational data are similar to an SQL query).

Fig. 5 shows an example of using file data query primitives. It is a query for a specific artwork and expresses the following meaning: there are two objects in the artwork; one is a dog and the features of its color are similar to those of picture dog.jpg, and the other is a house and the features of its shape are similar to those of picture house.jpg; the house is located on the left side of the dog.

```
USE DATABASE <database (data storage)>
SELECT [DISTINCT] <data>
FROM <database (data storage)>
[WHERE <search-condition (data features)>]
[GROUP BY <group_list> [HAVING <search-condition>]]
[ORDER BY <order_list> [ASC|DESC]]
```

Fig. 3 Relational data query primitives

```
USE DATABASE <database (data storage)>
SELECT [DISTINCT] <data>
FROM <database (data storage)>
[WHERE <search-condition (data features)>]
[GROUP BY <group_list> [HAVING <search-condition>]]
[ORDER BY <order_list> [ASC|DESC]]
<search-condition> ::= <condition> AND <condition>
<search-condition> ::= <condition> OR <condition>
<search-condition> ::= NOT <condition>
<search-condition> ::= <content>|<feature>|<spatial-temporal>
<content> ::= <data-name> IN <path>
<feature> ::= <function>|<attribute>|<similarity>
<spatial-temporal> ::= <path><temporal><spatial>[st]
```

Fig. 4 File data query primitives

Table 3 Explanation of the file query primitives

Keyword	Description
<search-condition>	Query conditions
<content>	Content information
<feature>	Feature information
<function>	Feature function
<attribute>	Data attribute
<similarity>	Similarity between entities
<temporal>	Time segment
<spatial-temporal>	Spatiotemporal information
<spatial>	Spatial segments
<path>	Data path
<st>	Azimuth axis or time axis

```
SELECT d.image
FROM DRAWINGS d, ARTISTS a
WHERE a.id=d.art_id
AND a.name='Xiao Ming'
AND dog_num=1, house_num=1 IN d.image.objects
AND similarity(dog, 'dog.jpg') AND similarity(house, 'house.jpg')
AND color(dog, 'dog.jpg')>0.9 AND shape(house, 'house.jpg')>0.8
AND dog LEFT house
```

Fig. 5 An example of using file data query primitives

As shown in Fig. 6, when D receives q, he/she should judge it to determine whether it complies with the access control policy. The judging of this part is based mainly on the following:

1. Judge the origin of q: check whether q is sent by an authorized user.

2. Check the query of q: judge whether all datasets can be obtained by combining all local application queries q. If not, continue with the following operations.

3. Judge whether q complies with the access rules in DDes.

The access rules in DDes define what data can be accessed and what data cannot be. For example, in the employee information table of the relational database, the user's name belongs to privacy and cannot be accessed at the same time. Salary and department cannot be accessed at the same time. Name and diagnosis-record or name and disease involve personal privacy and cannot be accessed.

If the check of q is not qualified, D will not return any decoder; otherwise, D uses function ϕ_3 to decrypt the private part of user application data from SMT_D^+ and reassemble it into a new $\mathrm{SMT}_{D^*}^+$. Then D uses function ϕ_4 to encapsulate the decoder $\otimes$, including $\mathrm{SMT}_{D^*}^+$, public parameters, and access control policies of data transmission. The decoder provides only the necessary interface for users to obtain the data slots they need.

We need to ensure that U cannot obtain the contents of the decoder through other means. Some basic indicators for encapsulating $\mathrm{SMT}_{D^*}^+$ include: (1) it should be invisible and unable to be disassembled; (2) it should be small enough to be placed on the blockchain; (3) it should be able to check whether the new query of data meets the policy. When the above encapsulation indicators are met, we can use a smart contract to act as a decoder.

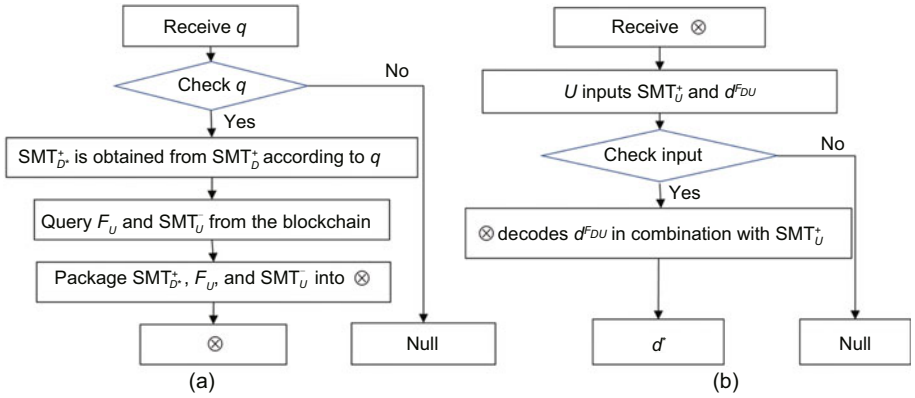

Fig. 6 Access control: (a) D phase; (b) U phase

q: query statement; d^F: data object with fingerprint F embedded; F_X: fingerprint of user X; SMT_X^-: public part of SMT_X; SMT_X^+: private part SMT_X; SMT: symbol mapping table; $\otimes$: encapsulated decoder

However, considering that smart contracts may have data privacy security problems, data parameters during contract execution could be disclosed. We find some particular solutions from the literature, and summarize them in the following categories:

1. Split contract (Kosba et al., 2016; Kalodner et al., 2018; Li et al., 2019). The contract for designing sensitive information is a private contract or an off-chain contract, and is not disclosed to the public.

2. Define smart contract language (Steffen et al., 2019; Baumann et al., 2020). Permission control is performed on variables, functions, and other elements of sensitive information designed in the contract.

3. Build a smart contract execution framework (Yan et al., 2020). Smart contracts are encrypted, decrypted, and executed in combination with a trusted execution environment (TEE).

In RCDS, we use the certification function and the black box nature of the TEE to have smart contracts executed securely. The core idea of the TEE is to build a hardware security area, and data are calculated only in the security statement to ensure their confidentiality and integrity. The running state of a smart contract in the TEE is trusted and cannot be obtained by the outside world, to ensure parameter safety in execution processes.

As shown in Fig. 6, when U receives the decoder sent by D, he/she will input his/her SMT_U^+ and $d^{F_{DU}}$. Then, the decoder uses $d^{F_{DU}}$ to check whether the user embeds his/her fingerprint in d^{F_D}. If the check is qualified, the decoder will return d^*. To this end, we design Algorithm 5, and implement it on the blockchain as a template for a smart contract for authorized users to query data. The first step to ensure query security is to check whether a user embeds his/her own fingerprint in d^{F_D} through ϕ_6. This largely prevents unauthorized users from using the decoder. After passing the ϕ_6 check, the decoder uses SMT_U^+ and SMT_{DU}^+ to convert $d^{F_{DU}}$ into d^{F_D}, and then recovers d^* from d^{F_D} by combining $\otimes$ and access control policies.

Algorithm 5 Decoder (ϕ_5)

Input: SMT_U^+, $d^{F_{DU}}$
Output: d^*
Obtain SMT_U^-, F_U, and q from blockchain
Obtain $\mathrm{SMT}_{D^*}^+$ and ac from D
/* ac: access control */
if $\phi_6(d^{F_{DU}}) =$ negative and !ac **then**
  return null
else
  for each $p \in d^{F_{DU}}$ **do**
    add $\mathrm{SMT}^-|_p$ to d^F
  end for
  for each $i \in d^F$ **do**
    if $i \in \mathrm{SMT}_{D^*}^+$ **then**
      $s \leftarrow \{s | s \in \mathrm{SMT}_{D^*}^+|_i\}$
      add s to d^*
    end if
  end for
  return d^*
end if

4.5 Blockchain network

Blockchain network is important for prompting the aforementioned fingerprint identification and

access control to come true, and should ultimately help realize the workflow shown in Fig. 1 and the data distribution model shown in Fig. 7. We design the blockchain network mainly in two parts: data model and consensus mechanism.

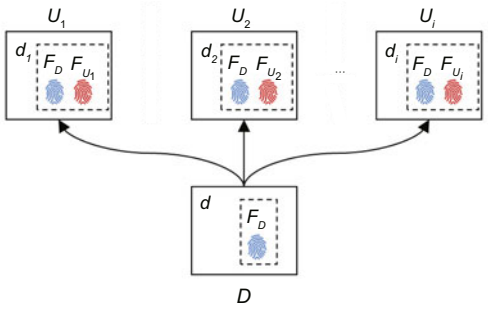

Fig. 7 Data distribution pattern for right confirmation

To collect data-sharing records correctly, we design the transaction chain data structure shown in Fig. 8 based on the following ideas: first, D creates an initial transaction txn1 which registers the data to be shared. At the same time, D embeds the fingerprint in the original data to form d^{F_D}, generates the corresponding DDes, and writes $h(\mathrm{SMT}_D^-)$, F_D, DDes, and $h(d^{F_D})$ into txn1. When data sharing is required, D can transfer d^{F_D} and DDes to any authorized user U. Next, when U receives the data d^{F_D}, he/she will execute SMC to obtain data $d^{F_{DU}}$, which contains his/her fingerprint. Meanwhile, q, F_U, and the hash digest of SMT_U^- are written into a transaction txn2. Finally, for each upcoming query q from any of the authorized users, the distributor will create a uniquely corresponding decoder $\otimes$ and write its digest into a transaction txn3, so that the authorized user can access the data object. These transactions are linked together in chronological order to constitute a transaction chain. Different transaction chains are intertwined with the blockchain. The transactions in each transaction chain transmit consensus in the network through the shielded pool (Kappos et al., 2018), and are then packaged into different blocks. From the above transaction process, we can find that every time an authorized user wants to access a data object, he/she applies to the distributor for access permission and this behavior will be verified on chain. Before the digest of the data object is uploaded to the blockchain, a decoder has had SMT_U^- and F_U encapsulated in the case where some attackers use fake parameters to defraud the distributor to obtain permission to access data.

For the consensus mechanism, we adopt the cascade consensus protocol (CCP) reported in our previous works (Wang L et al., 2020, 2021). CCP organically coordinates the periodic data transmission through a consensus process, which makes the data transmission undeniable. Its working mode meets the requirements of the data transmission part of RCDS.

We use Spring Boot to build a simulation blockchain platform (http://www.hbusoftsec.org.cn/files/rcds_bc.zip), which employs the above data model and consensus protocol. Some simulations that we discuss later in this study are conducted on this platform.

5 Model analysis

In this section, we present a theoretical analysis of RCDS effectiveness in DRC and attack resistance.

5.1 Non-repudiation of RCDS

Before U accesses the shared data copy, he/she must perform SMC, which will inevitably leave traces on the blockchain. Suppose that the suspected data d^F have been captured by A. A can perform traitor tracing using the following steps:

1. Read $\left(\mathrm{SMT}_U^-, F_U\right)$ from blockchain BC;
2. $\mathrm{Tester}_U = \phi_5\left(d^F, \mathrm{SMT}_U^-, F_U\right)$;
3. If Tester_U is positive, then U will be reasonably charged as a traitor;
4. Read $\left(\mathrm{SMT}_D^-, F_D\right)$ from BC;
5. $\mathrm{Tester}_D = \phi_5\left(d^F, \mathrm{SMT}_D^-, F_D\right)$;
6. If Tester_D is positive, then D will be identified as the owner of the data.

Then, the non-repudiation of data sharing with RCDS is analyzed as follows:

1. Both tasks of accessing data and tracing traitors are forced to obtain parameters from the blockchain. Doing so can form two-way containment of the misconduct of both D and U. If U provides fake SMT_U^- or F_U, q of U will fail to pass the censorship; if D provides fake SMT_D^- or F_D, there will be no way for D to authenticate the ownership of d. In the interests of both parties, the best strategy is to honestly abide by the rules of data delivery.

2. It is impossible to guess the fingerprints from d^F and SMT^-. d^F is actually a kind of cipher text of d because d and its SMT are separate from each

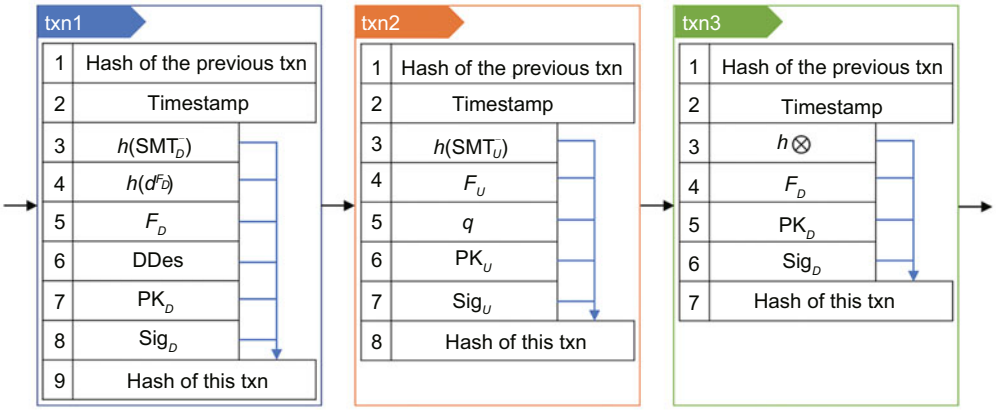

Fig. 8 Transaction chain over blockchain

txn: transaction; PK_X: public key of user X; Sig_X: signature of X; F_X: fingerprint of X; DDes: data description; d^F: data object with fingerprint F embedded; q: query statement; $h(\mathrm{SMT}_X^-)$: digest of the public part SMT_X; $h\otimes$: digest of a decoder

other. If the identity of D or U is unknown, the fingerprints in d^F are completely imperceptible. The invisibility of fingerprints leaves no room for traitors to deny their misbehavior.

3. D cannot frame U. If D intends to frame U, he/she has to use d^{F_D} and SMT_U^- to forge $d^{F_{DU}}$. However, such forgery is impossible in RCDS. First, $h(\mathrm{SK})$ customizes θ. Then, χ adds to the random redundancy of plain codes during the generation of SMTs (lines 14–20 in Algorithm 1), which makes θ more difficult to guess. Therefore, D is unable to forge $d^{F_{DU}}$ at an acceptable cost.

4. U cannot frame D. If U intends to frame D, he/she must either delete his/her fingerprint from $d^{F_{DU}}$ or publish a fake SMT_U^-. However, deleting fingerprints from a data object will only make the data object unusable. Meanwhile, the genuine SMT_U^- has been immutably recorded on the blockchain, and only this one can be used to unlock the data object in decoder $\otimes$. Therefore, U is not able to frame D by breaking fingerprints or undermining the synchronization between fingerprints and SMTs.

5. U cannot frame other users. If U intends to frame another user W, he/she has to embed W's fingerprint in d^{F_D} with greater strength. Even if W's fingerprint is embedded in d^{F_D} with ϕ_2, the fingerprint cannot be identified with ϕ_5. This is because the generation parameters of SMTs are fully managed by the blockchain network in a tamper-resistant manner. Moreover, even if W's fingerprint is successfully identified by mistake, W can also clear his/her suspicion with the help of D by disproving that d^{F_D} can be restored to d by using the corresponding SMT^- on chain.

5.2 Anti-piracy of RCDS

RCDS provides data sharing with distributed access control, which increases the difficulty and cost of data piracy. The effect of this access control is analyzed as follows:

1. D sends d^{F_D} and DDes to U through the blockchain.

2. U generates SMT_U^- according to the received d^{F_D} and its own F_U, and then generates the required query q according to DDes and sends it to D.

3. D sets $\otimes$ according to the query q and SMT_U^-, and sends it to U.

4. U uses its own SMT_U^+, $d^{F_{DU}}$, and $\otimes$ to query its application data.

5. U cannot obtain data that have not been applied for. If U wants to obtain unapproved data, it must pass $\otimes$, but D has set the judgment condition according to q in $\otimes$, so U cannot obtain unapproved data.

6. Other users cannot query data. If other users want to obtain data according to $\otimes$, they must have SMT_U^+ of U. However, SMT_U^+ is the private part SMT of U, so other users cannot obtain the corresponding data.

5.3 Surviving attacks

In this subsection, we analyze mainly the impact of various attacks on RCDS DRC capabilities.

5.3.1 Denial-of-service attacks

The risk of denial-of-service (DoS) attacks might exist in the blockchain network. However, the nodes located in the consortium of RCDS should all be approved, so there should be no motivation to actively commit DoS attacks. In addition, these nodes are usually protected by the consortium's defense in depth, and their safety should be ensured by systematic security mechanisms in all organizations of the consortium. As a data-sharing model, RCDS features mainly the ability of DRC, and many existing DoS defense methods can be used as supporting protection measures for RCDS.

5.3.2 Spoofing attacks

In RCDS, it is necessary to use the private keys of both parties to generate SMTs, and data must be partially obtained through authoritative decoders. D generates its own SMT and fingerprint based on its private key, so does U. Beyond that, the query q that U wants to execute is also associated with U's private key. Because these parameters are all immutably stored on the blockchain, the authenticity of each party is publicly verifiable, so most spoofing attacks can be avoided.

5.3.3 Sybil attacks

In RCDS, we adopt the CCP consensus protocol that we proposed in our early works (Wang L et al., 2020, 2021), which provides two ways to resist sybil attacks. One is to limit the way in which nodes join the network by forcing them to register valid data assets, that is, "First Share and Then Request." The other is to delete the suspicious nodes from the address list using cascaded message passing. To say the least, even if an RCDS system were hit by a sybil attack, the confidentiality of shared data and the privacy of honest nodes would not be compromised, because the current and future information that reaches consensus on blockchain is always desensitized and separated from the raw data.

5.3.4 Eclipse attacks

First, an eclipse attack is unlikely to succeed in a consortium blockchain network. Even if it happened, the victim nodes would simply be quarantined. The CCP consensus protocol we adopt in RCDS provides a fault tolerance rate of nearly $1/2$. This means that honest nodes can still reach consensus as long as they are more than half in number. Moreover, the node detection of CCP can find failed nodes agilely during at most one consensus round and exclude them from the network, thus preventing eclipse attacks from continuing.

5.3.5 Replay attacks

CCP, again, is essentially a fork-free consensus protocol. In each stage of CCP, the journey of a transaction begins and ends only on both parties of the transaction, while other nodes are responsible only for verifying and forwarding the transaction. In this process, each node grows its own blockchain, the consistency of which is temporally independent of transactions. Therefore, there is no problem of processing multiple new blocks simultaneously in CCP networks, which means that the hard-fork will not occur. Of course, replay attacks that exploit the vulnerability of hard-fork will not work against RCDS either.

5.3.6 User identity security

To prevent user identities from being analyzed in a way, we use the shielded pool concept to hide the addresses of both parties. We put all the records of on-chain transactions in the shielded pool; that is, when a transaction is conducted, the addresses of both parties will be encrypted right after the transaction enters the shielded pool. In this way, the on-chain pseudonyms and information, such as SMT^{-} and fingerprints, do not reveal users' real identities. In fact, the scenarios we envision for RCDS are generally for consortiums, and the connection between a user's real identity and a pseudonym is handled by the off-chain administration of the consortium, which means that the nodes in the blockchain network should have been authorized and opened, and it is meaningless for an attacker to analyze identities.

6 Evaluation

From the above analysis, we find that the credibility of the RCDS model depends mainly on the acceptable performance of the fingerprint identification and the blockchain operations. Therefore, we conducted a group of simulations to evaluate

the performance of RCDS mainly in correctness, robustness, and efficiency, and assess the efficiency of the blockchain network in terms of delay and throughput.

6.1 Configuration

In this subsection, we introduce the configuration for the fingerprint identification and blockchain operation tests.

6.1.1 Configuration for fingerprinting simulations

We programmed a testbed (http://www.hbusoftsec.org.cn/files/rcds_fingerprint.zip) in a Java SE v1.8 environment to simulate different fingerprint identification processes and test the fingerprint identification algorithms of RCDS on a single server (CPU: x64, 2.6 GHz, 6 cores, 12 logic processors; RAM: 16 GB; HDD: 1 TB; SSD: 256 GB).

To be compatible with common blockchain technologies, the testbed adopted SHA-256 as the hash digest algorithm and ECDSA as the asymmetric encryption and signature scheme. The parameters used in the testbed were preset as follows:

The parameter τ refers to the storage space limitation of SMT^-; we first assume that it is 1 MB in the simulations. The parameter λ refers to the symbol length factor. Its significance is to embed enough fingerprints in the hidden code so that they cannot be easily erased. We first assume $\lambda = 1024$ in the simulations. The parameter γ refers to the number of fingerprints that can be embedded in each character. Here, we should make it no more than 1, because if it exceeds 1, which means one plain code corresponding to multiple fingerprints, it will be very easy to guess and erase these fingerprints. Allowing for this, we initially assume $\gamma = 0.3$ in the simulations.

The main performance metrics of the simulations include correctness, robustness, and efficiency. Correctness reflects the ability of RCDS to successfully identify data fingerprints. Robustness reflects the ability of RCDS to resist malicious data processing. Efficiency reflects RCDS's executive agility.

The assessment terms we will use throughout the evaluation are defined as follows:

1. true positive (TP): the number of times that $\phi_5\left(d^{F_X}, \mathrm{SMT}^-, F_X, \cdots\right) = \text{positive}$;

2. false positive (FP): the number of times that $\phi_5\left(d^{F_X}, \mathrm{SMT}^-, F_Y, \cdots\right) = \text{positive}\,(X \neq Y)$;

3. true negative (TN): the number of times that $\phi_5\left(d^{F_X}, \mathrm{SMT}^-, F_Y, \cdots\right) = \text{negative}\,(X \neq Y)$;

4. false negative (FN): the number of times that $\phi_5\left(d^{F_X}, \mathrm{SMT}^-, F_X, \cdots\right) = \text{negative}$;

5. number of positive samples (P): P=TP+FN;

6. number of negative samples (N): N=TN+FP.

6.1.2 Configuration for blockchain simulations

We built a blockchain network (http://www.hbusoftsec.org.cn/files/rcds_bc.zip) by using the spring boot framework (JDK version: 1.8) for the performance testing. Then, we ran it in a server machine (CPU: Intel Xeon Platinum 8269CY Cascade Lake, 2.5 GHz, 12 cores; bandwidth: 1 Gb/s; memory: 4 GB; ESSD: 40 GB) by instantiating six containers and using multi-thread programming to simulate the communication between peer nodes.

6.2 Correctness of fingerprint identification

We measured the ability of RCDS to successfully identify fingerprints from data objects.

1. Metrics. Accuracy reflects how correct RCDS is. Precision and Sensitivity metrics inversely correlate with the false alarm rate and the missed alarm rate, respectively. These three metrics are defined as

$$\text{Accuracy} = \frac{\mathrm{TP}+\mathrm{TN}}{P+N},$$
$$\text{Precision} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}},$$
$$\text{Sensitivity} = \frac{\mathrm{TP}}{P}.$$

2. Settings. The fingerprinting effects of RCDS under different parameters were observed in the simulations. We simulated random datasets, in which the values of ε and ς were variable in the test. We set the parameters for this test as follows: d, randomly generated; number of users, 50; $|d|$, 1 MB; ε, 0.2–0.7; ς, 10–16. The test was calculated 10 times and the results were averaged.

3. Results. The results of the test are shown in Fig. 9.

4. Discussion. As shown in Fig. 9, the horizontal axis indicates the variation of ε, the vertical axis shows the values of the observed metrics, and the four subgraphs show the results for different values of ς.

It is easy to see that RCDS has perfect sensitivity when identifying fingerprint on randomly

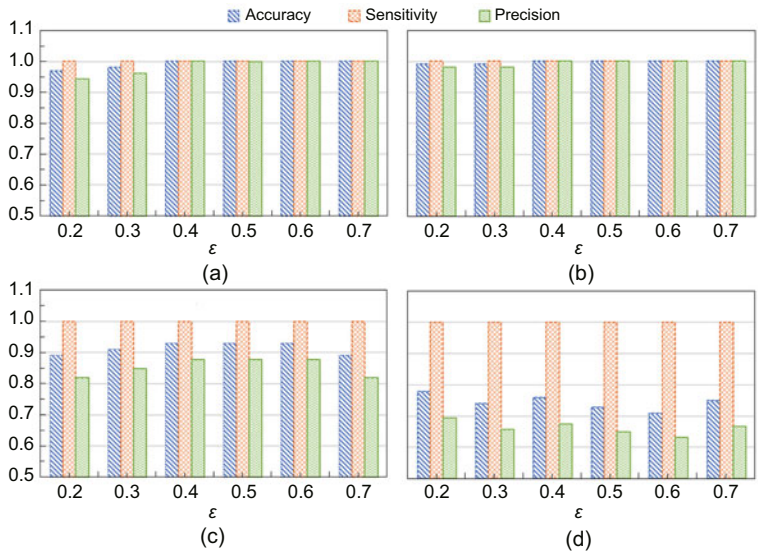

Fig. 9 Correctness of RCDS: (a) ς=10; (b) ς=12; (c) ς=14; (d) ς=16

generated datasets, free of malicious processing. In terms of Accuracy and Precision, when ς was set to a maximum of 12, the results were generally better than those at $\varsigma = 14$, which means that constraining the threshold of backward verification within a measurable range can ensure that the system works correctly. We found that when $\varepsilon \geq 0.4$, the fingerprint identification rate is as high as 100%, which gives the baseline of ε in forward verification.

To sum up, the simulation results showed that RCDS can correctly identify users' fingerprints from encoded data which were free of malicious processing.

6.3 Robustness of fingerprint identification

We measured RCDS's ability to correctly verify fingerprints from data objects when there was potentially malicious data processing.

1. Metrics. Same as those in Section 6.2.

2. Settings. RCDS's ability was evaluated in the simulations to resist the potential presentation attacks described below.

(1) Deletion attack. An adversary may delete a few bytes from d^F in an attempt to make the fingerprints undetectable. In the simulations, one third of d^F were deleted at random.

(2) Swap attack. An adversary may swap some pairs of bytes in d^F to disarrange the order of the codes. In the simulations, we swapped every two adjacent bytes.

(3) Padding attack. An adversary may put some random bytes over d^F, trying to reduce the fingerprint recall rate. In the simulations, we appended noisy bytes to d^F to extend the size of d^F to $2|d^F|$.

(4) Negation attack. An adversary may negate some bytes in d^F to obfuscate the codes. In the simulations, we negated half of the bytes.

(5) Reversion attack. An adversary may reverse the order in which d^F is stored, seeking to desensitize the program to correct fingerprints. In the simulations, we completely reversed d^F.

We chose the above attack models because each of them represents a class of content processing. ϕ_5 of RCDS has a strong adaptability because it does not depend on the type of data content. For each attack model above, we observed the influence of $\langle \varepsilon, \varsigma \rangle$ ($\varepsilon \in \{0.4, 0.5\}, \varsigma \in \{12, 14\}$) on the fingerprinting effect. The test was calculated 10 times, and the results were averaged.

3. Results. The results of the test are shown in Fig. 10.

4. Discussion. As shown in Fig. 10, the horizontal axis lists the five attack types mentioned in the settings, the vertical axis shows the values of observed metrics, and the four subgraphs show the results based on different values of ε and ς.

It is clear that RCDS worked the best against those attacks when $\varepsilon = 0.5$ and $\varsigma = 14$, which tells us the threshold for making the system live in a malicious environment. It can be found that increasing

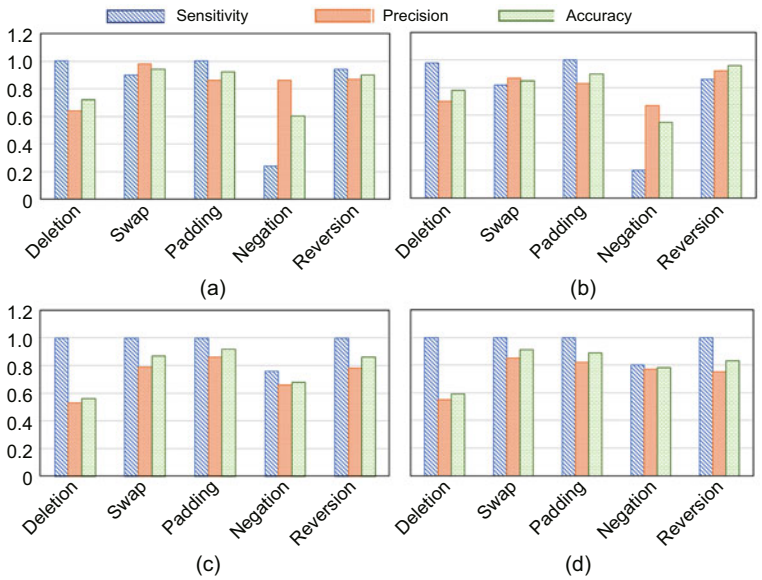

Fig. 10 Robustness of RCDS: (a) ε=0.4, ς=12; (b) ε=0.5, ς=12; (c) ε=0.4, ς=14; (d) ε=0.5, ς=14

ς will increase the sensitivity of RCDS to these five types of attacks. In addition, the effect of a negative attack was not as good when $\varsigma < 14$. The reason is that the negative attack essentially manipulates the plain code of d^F, while the forward verification phase of Algorithm 3 must be relatively rigorous to interpret the plain code. However, by raising ς, the recognition rate against this attack obviously increased. It is important to note that the effect of fingerprinting is generally not influenced by the volumes and types of data objects, because the target of RCDS is the byte sequences of the data objects. All the above results support the acceptable reliability of RCDS.

6.4 Efficiency of fingerprint identification

We measured the efficiency of RCDS in running SMT generation and fingerprint identification.

1. Metrics. The major performance costs of RCDS come from the activities of SMT generation and fingerprint identification. The symbol mapping generation runtime (SMRT) and the fingerprint identification runtime (FIRT) were logged when observing these two types of activities.

2. Settings. The inputs that are closely related to RCDS's efficiency include θ, η, and $|d|$. The impact of these inputs' changes on SMRT and FIRT was observed in the simulations through several tests. According to inequalities (1) and (3), we calculated the ranges of θ and η based on different assignments of $|d|$. Table 4 lists their values for this simulation. Each test was calculated 10 times, and the results were averaged.

Table 4 Settings for the efficiency simulation

Test number	$\lvert d\rvert$ (MB)	θ	η
1	1	35–60	10–30
4	8	280–475	10–30
5	16	562–952	10–30
6	32	1128–1898	10–30

3. Results. Figs. 11 and 12 show SMRT and FIRT, respectively.

4. Discussion. As shown in Figs. 11 and 12, the horizontal axis indicates the variation of the hidden code length η, the vertical axis shows the values of observed metrics, and the four subgraphs show the SMRT and FIRT with different amounts of data, and the series shown in each subgraph represents different values of θ.

As seen in the results, we can adjust the value of θ to fit any size of $|d|$, while the efficiency of RCDS will not be affected. In other words, the size of $|d|$ has almost no influence on the runtime, which indicates that RCDS is scalable in terms of data volume. At the same time, SMRT and FIRT decreased and tended to be stable with the increase of θ when the data volume was the same. The reason is that the possibility of symbol repeating in $\left|\mathrm{SMT}_D^+\right|$ decreases with the increase of θ.

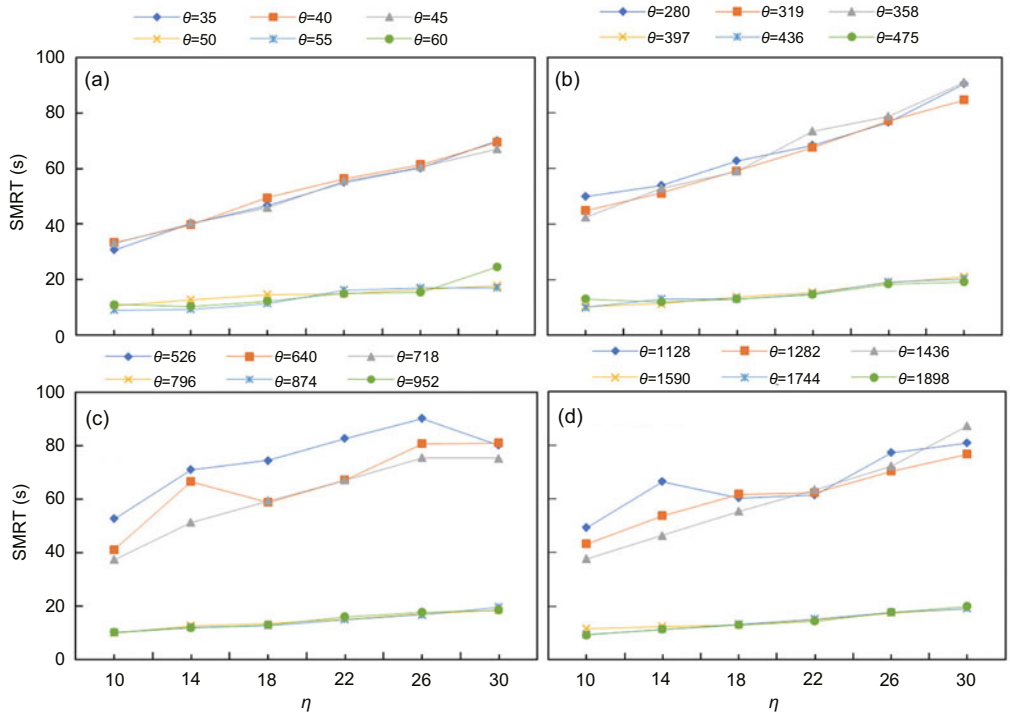

Fig. 11 Symbol mapping generation runtime (SMRT) of RCDS: (a) $|d|$=1 MB; (b) $|d|$=8 MB; (c) $|d|$=16 MB; (d) $|d|$=32 MB

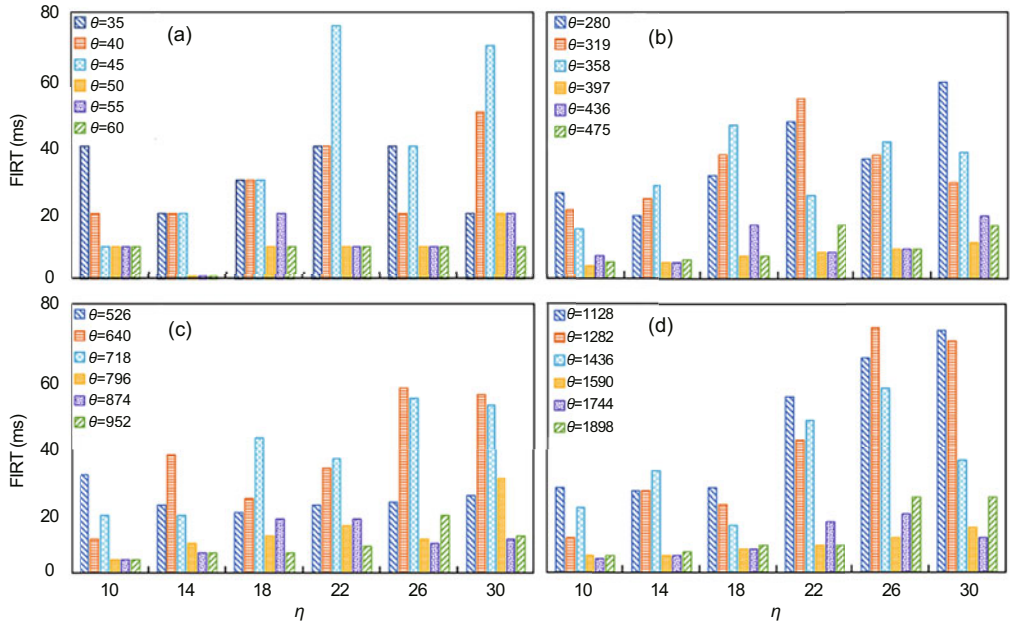

Fig. 12 Fingerprint identification runtime (FIRT) of RCDS: (a) $|d|$=1 MB; (b) $|d|$=8 MB; (c) $|d|$=16 MB; (d) $|d|$=32 MB

At the same time, when θ was constant, the results showed that a smaller value of η usually led to less runtime, as smaller hidden code lengths always require less searching effort.

With regard to performance expansion, the values of θ and η affected RCDS's storage and communication performance. From $|d| = \theta|S|$ and $\left|d^F\right| = \rho|S|$, we know that the larger the θ, the

smaller the $\left|d^F\right|$, and the less storage and communication pressure there will be.

In addition, in Fig. 12, we can notice that some values were abnormally large. This was caused by the backward verification of the fingerprint (Algorithm 4), and was consistent with the theoretical basis of RCDS's full fingerprint coverage strategy (i.e., $\varphi|F| = \eta|S|$).

6.5 Blockchain performance

We tested the average delay and throughput of the blockchain network when the RCDS model was working.

1. Metrics. The average delay indicates mainly how fast a single transaction is confirmed, and the throughput reflects the number of transactions completed per unit of time.

2. Settings. We set the numbers of nodes as $\{3, 4, 5\}$, and the number of transactions from 5000 to 25 000.

3. Results. The results are shown in Fig. 13.

4. Discussion. The duration of transaction generation and the time consumed by consensus are the main factors affecting blockchain efficiency. From Fig. 13, we learn that the throughput changed little with increased numbers of nodes and transactions, and that the average delay was within an acceptable range. This means that RCDS does not cost much to run if an appropriate blockchain network is deployed, and that DRC is quite feasible on such a model.

7 Conclusions and future work

In this work, we propose RCDS—a right-confirmable data-sharing model. By using SMC, RCDS encodes raw data in a non-distortion way, and thus is competent for DRC regardless of the types and volumes of shared data. By employing blockchain, RCDS imposes credible supervision on DRC through the whole network consensus. Furthermore, RCDS combines SMC and blockchain into a systematic mechanism, with which the data access can be fully under control during its sharing processes. Above features of RCDS make it possible to launch trusted traitor tracing and access control, better supporting the forensics on the acts of transaction repudiation and data piracy.

Of course, our work inevitably has some limitations, and further research and expansion are needed. First, RCDS does not provide a fingerprint extraction function, which might further improve the credibility of fingerprint identification. Second, it is necessary to design a unified and effective access control strategy for different data content types, which can make the decoder encapsulation more secure. Third, a more complete user identity privacy protection scheme is required to ensure the security of user identities. Finally, the encapsulation strategy in this model is implemented by smart contracts, which always need closer check in security. We hope that dealing with the above issues will lead to the emergence of more effective DRC data-sharing models.

Contributors

Liang WANG designed the research. Shunjiu HUANG conducted the simulations and drafted the paper. Lina ZUO processed the data and helped organize the paper. Jun LI performed the formal analysis. Wenyuan LIU supervised the research. Liang WANG revised and finalized the paper.

Compliance with ethics guidelines

Liang WANG, Shunjiu HUANG, Lina ZUO, Jun LI, and Wenyuan LIU declare that they have no conflict of interest.

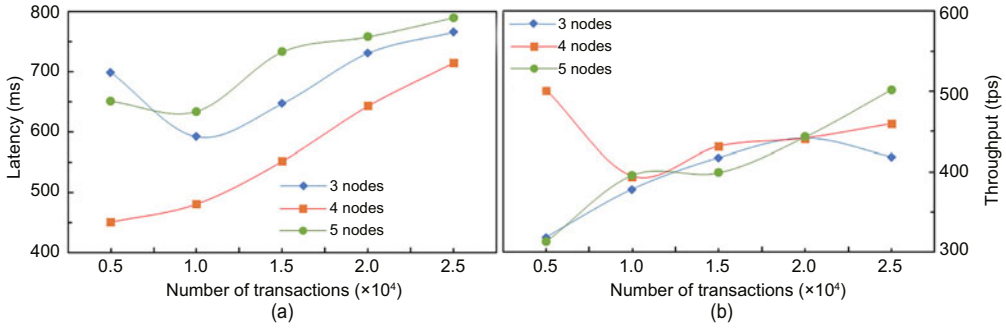

Fig. 13 Comparison of average latency (a) and throughput (b) (tps: transactions per second)

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Ali M, Reaz R, Gouda M, 2016. Two-phase nonrepudiation protocols. Proc 7th Int Conf on Computing Communication and Networking Technologies, Article 22. https://doi.org/10.1145/2967878.2967903

Barni M, Bartolini F, 2004. Data hiding for fighting piracy. *IEEE Signal Process Mag*, 21(2):28-39. https://doi.org/10.1109/MSP.2004.1276109

Baumann N, Steffen S, Bichsel B, et al., 2020. zkay v0.2: practical data privacy for smart contracts. https://arxiv.org/abs/2009.01020

Cao ZH, Zhao L, 2021. A design of key distribution mechanism in decentralized digital rights management based on blockchain and zero-knowledge proof. Proc 3rd Int Conf on Blockchain Technology, p.53-59. https://doi.org/10.1145/3460537.3460556

Chen F, Wang JH, Li JQ, et al., 2022. TrustBuilder: a nonrepudiation scheme for IoT cloud applications. *Comput Secur*, 116:102664. https://doi.org/10.1016/j.cose.2022.102664

Coffey T, Saidha P, 1996. Non-repudiation with mandatory proof of receipt. *ACM SIGCOMM Comput Commun Rev*, 26(1):6-17. https://doi.org/10.1145/232335.232338

Ersoy O, Genç ZA, Erkin Z, et al., 2021. Practical exchange for unique digital goods. Proc IEEE Int Conf on Decentralized Applications and Infrastructures, p.49-58. https://doi.org/10.1109/DAPPS52256.2021.00011

Frattolillo F, 2017. Digital copyright protection: focus on some relevant solutions. *Int J Commun Netw Inform Secur*, 9(2):282-293. https://doi.org/10.17762/ijcnis.v9i2.2425

Gai KK, Choo KKR, Zhu LH, 2018. Blockchain-enabled reengineering of cloud datacenters. *IEEE Cloud Comput*, 5(6):21-25. https://doi.org/10.1109/MCC.2018.064181116

Ganesh SM, Pandi V, Deborah LJ, et al., 2017. Attacks on the anti-collusion data sharing scheme for dynamic groups in the cloud. Proc Int Conf on Security, Privacy and Anonymity in Computation, Communication and Storage, p.457-467. https://doi.org/10.1007/978-3-319-72395-2_42

Giacomelli I, Madsen J, Orlandi C, 2016. ZKBoo: faster zero-knowledge for Boolean circuits. Proc 25th USENIX Conf on Security Symposium, p.1069-1083.

Gong JQ, Lin SF, Li JW, 2019. Research on personal health data provenance and right confirmation with smart contract. Proc IEEE 4th Advanced Information Technology, Electronic and Automation Control Conf, p.1211-1216. https://doi.org/10.1109/IAEAC47372.2019.8997930

Huckle S, White M, 2017. Fake news: a technological approach to proving the origins of content, using blockchains. *Big Data*, 5(4):356-371. https://doi.org/10.1089/big.2017.0071

Kalodner H, Goldfeder S, Chen XQ, et al., 2018. Arbitrum: scalable, private smart contracts. Proc 27th USENIX Security Symp, p.1353-1370.

Kappos G, Yousaf H, Maller M, et al., 2018. An empirical analysis of anonymity in Zcash. https://arxiv.org/abs/1805.03180

Kosba A, Miller A, Shi E, et al., 2016. Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. Proc IEEE Symp on Security and Privacy, p.839-858. https://doi.org/10.1109/SP.2016.55

Li C, Palanisamy B, Xu RH, 2019. Scalable and privacy-preserving design of on/off-chain smart contracts. Proc 35th Int Conf on Data Engineering Workshops, p.7-12. https://doi.org/10.1109/ICDEW.2019.00-43

Lin C, Luo M, Huang XY, et al., 2022. An efficient privacy-preserving credit score system based on noninteractive zero-knowledge proof. *IEEE Syst J*, 16(1):1592-1601. https://doi.org/10.1109/JSYST.2020.3045076

Okonkwo IE, 2021. NFT, copyright and intellectual property commercialization. *Int J Law Inform Technol*, 29(4):296-304. https://doi.org/10.1093/ijlit/eaab010

Pan H, You XM, Liu S, et al., 2021. Pearson correlation coefficient-based pheromone refactoring mechanism for multi-colony ant colony optimization. *Appl Intell*, 51(2):752-774. https://doi.org/10.1007/s10489-020-01841-x

Parno B, Howell J, Gentry C, et al., 2016. Pinocchio: nearly practical verifiable computation. *Commun ACM*, 59(2):103-112. https://doi.org/10.1145/2856449

Qian P, Liu ZG, Wang X, et al., 2019. Digital resource rights confirmation and infringement tracking based on smart contracts. Proc 6th Int Conf on Cloud Computing and Intelligence Systems, p.62-67. https://doi.org/10.1109/CCIS48116.2019.9073733

Saini A, Zhu QY, Singh N, et al., 2021. A smart-contract-based access control framework for cloud smart healthcare system. *IEEE Int Things J*, 8(7):5914-5925. https://doi.org/10.1109/JIOT.2020.3032997

Sifah EB, Xia Q, Xia H, et al., 2021. Selective sharing of outsourced encrypted data in cloud environments. *IEEE Int Things J*, 8(18):14141-14155. https://doi.org/10.1109/JIOT.2021.3068226

Steffen S, Bichsel B, Gersbach M, et al., 2019. zkay: specifying and enforcing data privacy in smart contracts. Proc ACM SIGSAC Conf on Computer and Communications Security, p.1759-1776. https://doi.org/10.1145/3319535.3363222

Sun XQ, Yu FR, Zhang P, et al., 2021. A survey on zero-knowledge proof in blockchain. *IEEE Netw*, 35(4):198-205. https://doi.org/10.1109/MNET.011.2000473

Wang HL, Tian YL, Yin X, 2018. Blockchain-based big data right confirmation scheme. *Comput Sci*, 45(2):15-19, 24 (in Chinese). https://doi.org/10.11896/j.issn.1002-137X.2018.02.003

Wang L, Liu JY, Liu WY, et al., 2020. Blockchain-based diversion-point system for balancing customer flow in shopping mall. *Symmetry*, 12(12):1946. https://doi.org/10.3390/sym12121946

Wang L, Liu JY, Liu WY, 2021. Staged data delivery protocol: a blockchain-based two-stage protocol for non-repudiation data delivery. *Concurr Comput Pract Exp*, 33(13):e6240. https://doi.org/10.1002/cpe.6240

Wang L, Li J, Zuo LN, et al., 2022. T-tracer: a blockchain-aided symbol mapping watermarking scheme for traitor tracing in non-repudiation data delivery. Proc 4th ACM Int Symp on Blockchain and Secure Critical Infrastructure, p.23-34. https://doi.org/10.1145/3494106.3528674

Wang S, Yang M, Ge TJ, et al., 2022. BBS: a blockchain big-data sharing system. Proc IEEE Int Conf on Communications, p.4205-4210. https://doi.org/10.1109/ICC45855.2022.9838666

Wu ZD, 2009. A Multimedia Query Language and its Query Processing. PhD Thesis, Huazhong University of Science and Technology, Wuhan, China (in Chinese).

Yan Y, Wei CZ, Guo XP, et al., 2020. Confidentiality support over financial grade consortium blockchain. Proc ACM SIGMOD Int Conf on Management of Data, p.2227-2240. https://doi.org/10.1145/3318464.3386127

Zaghloul E, Zhou K, Ren J, 2020. P-MOD: secure privilege-based multilevel organizational data-sharing in cloud computing. *IEEE Trans Big Data*, 6(4):804-815. https://doi.org/10.1109/TBDATA.2019.2907133

Zha C, Yin H, Yin B, 2020. Data ownership confirmation and privacy-free search for blockchain-based medical data sharing. Proc 2nd Int Conf on Blockchain and Trustworthy Systems, p.619-632. https://doi.org/10.1007/978-981-15-9213-3_48

Zhang LY, Zheng YF, Weng J, et al., 2020. You can access but you cannot leak: defending against illegal content redistribution in encrypted cloud media center. *IEEE Trans Depend Secur Comput*, 17(6):1218-1231. https://doi.org/10.1109/TDSC.2018.2864748

Zhang R, Zhang L, Choo KKR, et al., 2023. Dynamic authenticated asymmetric group key agreement with sender non-repudiation and privacy for group-oriented applications. *IEEE Trans Depend Secur Comput*, 20(1):492-505. https://doi.org/10.1109/TDSC.2021.3138445

Zhao WB, Jiang CF, Gao HH, et al., 2021. Blockchain-enabled cyber-physical systems: a review. *IEEE Int Things J*, 8(6):4023-4034. https://doi.org/10.1109/JIOT.2020.3014864

Zhu LH, Wu YL, Gai KK, et al., 2019. Controllable and trustworthy blockchain-based cloud data management. *Fut Gener Comput Syst*, 91:527-535. https://doi.org/10.1016/j.future.2018.09.019

Zhu ZM, Jiang R, 2016. A secure anti-collusion data sharing scheme for dynamic groups in the cloud. *IEEE Trans Parall Distr Syst*, 27(1):40-50. https://doi.org/10.1109/TPDS.2015.2388446