SCIENCE CHINA
Information Sciences

March 2020, Vol. 63 130101:1–130101:13
https://doi.org/10.1007/s11432-018-9781-0

• RESEARCH PAPER •
Special Focus on Security and Privacy in Blockchain-based Applications

# Blockchain-based multiple groups data sharing with anonymity and traceability

Hui HUANG[1,2], Xiaofeng CHEN[1*] & Jianfeng WANG[1]

[1]*State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an* 710071*, China;*
[2]*College of Computer, Minnan Normal University, Zhangzhou* 363000*, China*

Received 31 October 2018/Accepted 23 January 2019/Published online 9 August 2019

**Abstract** Group data sharing enables information sharing between multiple parties for cooperative purposes. However, the existing schemes only consider scenarios in which all parties in the same organization want to share data. Achieving secure data sharing between users of different groups is also a relevant research issue. In this paper, we propose a blockchain-based data sharing scheme for multiple groups with anonymity and traceability. Owing to the consortium blockchain technique, any user in the system can easily verify the validity of the shared data without interacting with a third-party auditor. Additionally, the proposed scheme can not only enable data sharing between different groups with enhanced security anonymously but also achieve traceability and non-frameability.

**Keywords** multiple groups, data sharing, blockchain, anonymity, traceability

**Citation** Huang H, Chen X F, Wang J F. Blockchain-based multiple groups data sharing with anonymity and traceability. Sci China Inf Sci, 2020, 63(3): 130101, https://doi.org/10.1007/s11432-018-9781-0

## 1 Introduction

A cloud storage service enables enterprises and individuals to have a comparable, low-cost, and scalable platform service. It provides users with data storage services by integrating numerous storage devices in a network. Thus, users can securely share their data through the cloud storage service provided by the cloud service providers [1–6]. Increasing applications of data sharing have emerged in recent years, for instance, iCloud and Google Docs.

Group data sharing is a specific case that enables information sharing between multiple users in a group for cooperative purposes. It has many practical applications, such as electronic health networks, wireless body area networks, and electronic literature in libraries. With the rapid development of cloud storage, realizing group data sharing through cloud storage services has attracted considerable attention in the academic community [7]. In the existing research, there are two methods to realize data sharing between users in a group: one-to-many and many-to-many. The first method is suitable for cases where there is only one data owner and only authorized users can access his data resource. The second method allows authorized users to access the data resources of different data owners within the same group. However, the existing literatures consider only the scenario that all the users in the same group want to share data. In this paper, we consider a realistic scenario in which multiple users of different groups want to share their data, e.g., multiple academic organizations intending to share their research results or discoveries, multiple medical institutions wanting to establish a shared medical database, and multiple universities

* Corresponding author (email: xfchen@xidian.edu.cn)

 

wanting to share their educational resources. In this scenario, users of different groups can realize data sharing by storing data in the cloud. However, the cloud is "semi-trusted", and users are concerned about the unpredictable security problems caused by the loss of data control. Moreover, the data stored in the cloud are from different groups. Users of different groups do not trust the reliability of the data from other organizations. In addition, users would prefer anonymous data sharing to protect privacy.

Therefore, to realize secure data sharing between different groups under the cloud computing environment, the following challenges should be considered. First, the data are from different groups, and users of different groups do not trust the reliability of the data from other groups. Therefore, the users should verify the integrity and validity of the data stored in the cloud server. The existing methods for data verification are all based on third-party audits, namely, a third-party auditor is introduced into the process of data auditing. However, the third party is honest but curious. It may gather personal information from the audit data. How to solve the trust problem of the third party should be considered. Second, multiple groups are working together to build a shared database. In contrast to the one-group model, the managers of these organizations have equal rights in the data management. Accordingly, how to design a scheme for the case of multiple groups remains a considerable challenge. Finally, the privacy of users should be protected in the system. To achieve this goal, users should be anonymous. When a dispute occurs, the manager of the group can reveal the true identity of the user. However, only a single entity has the complete privilege of tracking the identity, which leads to the problem that honest group members may be framed and dishonest members may be shielded.

In addition, blockchain is an emerging technology in the field of information. It has the characteristics of decentralization, openness, autonomy, and tamper-resistance. The data recorded on a blockchain cannot be tampered with and revised. A blockchain can be regarded as a decentralized trusted third party. Therefore, the decentralization of a blockchain provides a viable solution for building security schemes without a trusted third party. Concurrently, the blockchain technology enables users to construct a shared, distributed, and fault-tolerant database. It is also very suitable for designing data sharing schemes. In this paper, we propose a blockchain-based data sharing scheme for multiple groups with anonymity and traceability.

### 1.1 Main contributions

• We present a specific data sharing scheme for multiple groups based on the concept of consortium blockchain. In our construction, the verification information for auditing is recorded in a tamper-resistant database that is maintained by all the participants using the consortium blockchain technology. The users in the system can easily verify the integrity of the shared data without interacting with a third party.

• We adopt a mechanism where we store the actual data in the storage server instead of in a blockchain in our data sharing structure. Only the verification information is stored in a blockchain, which is a public ledger that can be read by each user in the system. Therefore, the actual data are imperceptible to anyone in the system. We also use group signature techniques to achieve anonymous information exchange in the public blockchain. Thus, the privacy of user identities is protected.

• We combine the concept of a group signature with a blockchain to avoid excessive power to the group manager. When a dispute occurs, the group manager can reveal the identity of the data owner. Moreover, fairness can be guaranteed in the process of solving a dispute, i.e., honest users cannot be framed and dishonest users cannot be shielded.

### 1.2 Related work

Group data sharing is a specific case that enables information sharing between multiple users for cooperative purposes. Numerous data sharing schemes under the cloud environment have been proposed in [8, 9]. Wang et al. [1] proposed the first privacy-preserving public auditing method for data sharing in the cloud. The proposed scheme utilized a ring signature to construct a homomorphic authenticator. In the process of data verification, the public verifier did not know the identity information. However, the

method is an unconditional privacy protection scheme. The group signature technique has advantages in terms of anonymity and traceability. Based on the group signature technique, Liu et al. [3] proposed a secure scheme that supported anonymous data sharing in cloud computing. Wang et al. [4] designed a homomorphic authenticator based on a group signature and proposed a public auditing method for the shared data in the cloud with anonymity and traceability. Shen et al. [10] introduced a data sharing scheme with security and efficiency. They focused on the data sharing in the same group. Based on the symmetric balanced incomplete block design (SBIBD) and group signature technique, their solution could realize secure group data sharing anonymously. Concurrently, the security of the data could be protected.

To realize data security sharing, data integrity verification is an important security requirement [11,12]. Two types of solutions are used to solve the problem in the existing research. The two-party auditing model [13, 14] was the first to be used in the data auditing for a data sharing scheme. In this model, an independent auditing service is needed to solve the problem of data integrity verification. With the emergence of cloud storage services, data sharing can be realized by storing data in the cloud server. However, the two-party auditing model is unsuitable for the new scenario owing to its ineffectiveness. Ateniese et al. [15] proposed a scheme of the third-party auditing model for data integrity verification, called the provable data possession (PDP) model. In their scheme, they used homomorphic tags for data verification and realized data verification by a challenge-response protocol. However, they did not consider the problem of dynamic auditing. Subsequently, several extended schemes based on the provable data possession model were proposed to solve different problems [8, 9, 16–22]. Erway et al. [18] extended the concept of the provable data possession and proposed the concept of the dynamic PDP (DPDP) model. They presented two concrete schemes for the DPDP model. However, the proposed protocol could not support public algorithms. Wang et al. [23] presented a dynamic scheme based on the Merkle hash tree, which satisfied the requirements of public auditing.

Almost all the above solutions for data sharing are based on the third-party auditing model. However, the solution of introducing a third-party auditor leads to higher computational costs and communication overhead. Recently, blockchain has emerged as a novel technology [24–27] that enables users to construct a shared, distributed, and fault-tolerant database [28]. The decentralized property of a blockchain can be used to build an auditorless auditing scheme. Ghoshal et al. [29] proposed the first auditing mechanism without a third-party requirement, but they did not consider the privacy and security of the users. Blockchain-based data auditing can provide tamper-proof records and enable data accountability in the cloud. However, owing to its openness and transparency, each participant in the network can read all the information in the public ledger. Therefore, the privacy protection of user information should be considered. To the best of our knowledge, no researchers have studied multiple-group data sharing with privacy based on a blockchain.

### 1.3 Organization

The remainder of this paper is organized as follows. Some preliminaries are given in Section 2. The system model and design goals are presented in Section 3. The proposed scheme is described in Section 4. The security performance analysis is provided in Section 5. Finally, conclusion is made in Section 6.

## 2 Preliminaries

### 2.1 Group signature

Group signature, enabling any group member to sign a message in the name of the group, was first proposed by Chaum and van Heyst in 1991 [30]. In a group signature scheme, any group member can sign a message anonymously on behalf of the entire group. Similar to a general digital signature, a group signature is publicly verifiable and can be verified by a single group public key. Moreover, the actual

identity of the signer in the system cannot be traced by the verifier. Only the group manager can identify the real signer. A group signature scheme should satisfy the following requirements:

• **Unforgeability.** No one can generate a valid group signature except the members of the group.

• **Anonymity.** Given a group signature, determining the identity of the signer is computationally infeasible for anyone except the manager of the group.

• **Traceability.** The group manager can trace the real identity of a malicious user when a dispute occurs.

• **Unlinkability.** Without opening the group signature, it is difficult to distinguish whether two different signatures are made by the same group member.

• **Non-frameability.** No one, including the group manager, can generate a valid group signature in the name of other group members.

The advantageous property of a group signature is that it enables suitable anonymity in various scenarios. However, a group manager has excessive power in a group signature scheme. He manages all the group members, and only he can trace the real identity of a group member. In this case, only a single entity has the complete privilege of tracking identity, which leads to the problem that honest group members may be framed and dishonest members may be shielded.

### 2.2 Blockchain

A blockchain can be seen as a distributed database based on the blockchain technology. The blockchain technology refers to the combination of data exchange, processing, and storage technologies between multiple participants based on modern cryptography, distributed consistency protocol, point-to-point network communication technology, and smart contract programming language. The blockchain technology was first used for Bitcoin, introduced by Nakamoto in 2008 [31]. Currently, blockchain networks can be divided into three major categories: public blockchain, private blockchain, and consortium blockchain.

• **Public blockchain.** A public blockchain allows anyone to visit and send transaction requests to the blockchain network. Anyone (also called a miner) with an Internet connection can participate in the execution of a consensus protocol. Some well-known example applications are Bitcoin and Ethereum. A public blockchain is usually called a permissionless blockchain.

• **Private blockchain.** In a private blockchain network, participants can join when permitted by the network administrator. Participant and validator (miner) access is restricted. This type of blockchain is more suitable for internal applications within an organization, such as database management and auditing.

• **Consortium blockchain.** A consortium blockchain is a blockchain that requires registration permission. It is also called a permissioned blockchain. In contrast to a private blockchain, different organizations can join such a network. Each organization operates a node. In a consortium blockchain, the nodes participating in the blockchain are selected in advance, and there are usually good network connections between these nodes. Data on the blockchain are either public or internal. Therefore, a consortium blockchain is often said to be semi-decentralized. For example, numerous financial institutions have established consortium blockchains. Each institution runs a node, and confirmation from at least 10 of the institutions must be obtained to make each block effective. The blockchain can allow each institution to read, limit reading to consensus verification participants, or operate in a mixed way.

For instance, the root hash of the block and the application interface are open to the public and allow the outside world to query the blockchain data and blockchain status information. The connection between the participating nodes of the alliance chains is good, and the verification efficiency is high. The operation can be maintained at a minimum cost, and the transaction costs are reduced while simultaneously providing high-speed transaction processing. The expansibility is excellent, and the data can maintain privacy to some extent. However, the participating nodes can tamper with the data under the condition that a consensus is reached.

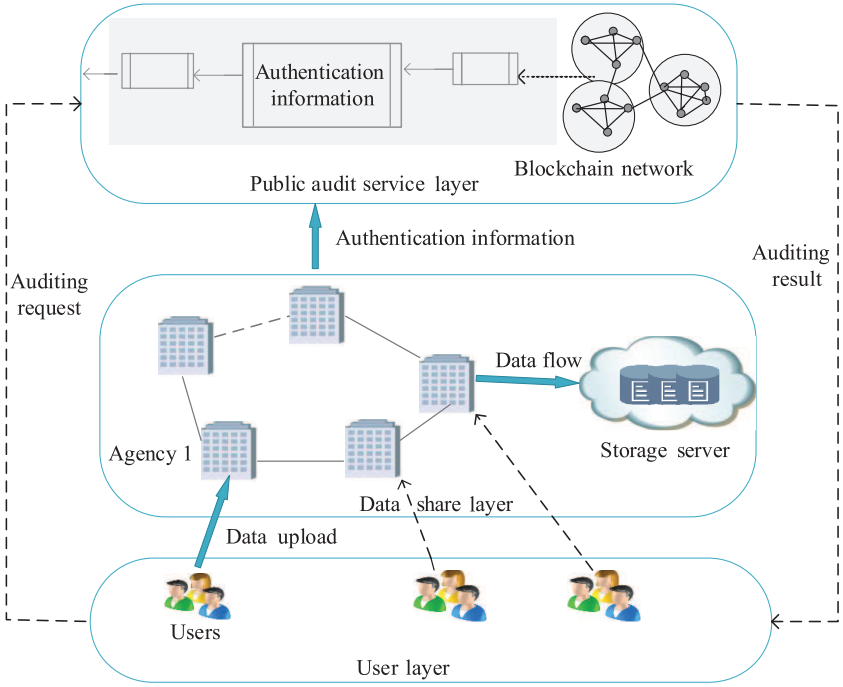


**Figure 1** (Color online) The system model of our scheme.

## 3 Blockchain-based multiple groups data sharing scheme

### 3.1 System model

In our architecture, we consider the following specific scenario. Multiple organizations jointly establish a shared database (e.g., academic organizations intending to share their research results or discoveries). They have equal rights in terms of data management and user management. In our system, we adopt a consortium blockchain to establish the data sharing scheme. Therefore, three entities are included in the system: agencies, users, and nodes.

(1) Agencies work together to provide data sharing services to users. We assume these agencies have the ability to store and manage the data of the local users. In our system, these agencies can be banks, research institutions, universities, and others. In addition, agencies are seen as the blockchain members in a consortium blockchain network. Agencies are semi-trusted parties in our scheme. Users within an agency are trusted (i.e., students trust their university), but users from other agencies are untrusted (i.e., different universities do not trust each other).

(2) Users are the data owners. They belong to different agencies and want to share data through the agencies.

(3) Nodes are controlled by the blockchain members. They run some distinct algorithms to maintain a public ledger that records all the verification information.

Figure 1 shows the system model of our scheme. We use a three-layer architecture to describe the implementation of our scheme. The first layer is the user layer. Users register on the agency and upload their shared data on it. The second layer is the data sharing layer, which consists of multiple agencies. This layer provides data sharing services to all the users. The function of this layer is to store and manage data. A more important function is that all the members work together to maintain a blockchain database, which provides a public audit service to the users in the third layer. The third layer is the public audit service layer. In this layer, a tamper-resistant database that records the public verification information for the users to verify the integrity and availability of the shared data is maintained by all the agencies. We assume that all the users and agencies can access the blockchain database. In our scheme, all the members can use a verification protocol to verify the integrity of the data without requiring a

trusted third party.

### 3.2 Security properties

Our proposed scheme should satisfy the following security properties.

• **Data confidentiality.** The shared data are not visible to the blockchain nodes. In our scheme, the blockchain nodes maintain a blockchain database that records the verification information of the shared data. The nodes should verify the correctness of the shared data, but they should not obtain real data.

• **Anonymity.** The data owners want to share their data without making their true identity public because the data may contain individual private information. To protect the user privacy, the data should be shared anonymously in the proposed scheme.

• **Traceability.** Under certain conditions, the group manager can trace the real identity of a malicious user when a dispute occurs.

• **Public verifiability.** All the users can verify the integrity of the shared data without requiring a trusted third party.

• **Non-frameability.** Fairness in the process of tracing should be guaranteed, i.e., honest users cannot be framed and dishonest users cannot be shielded.

### 3.3 The proposed scheme

In this section, we describe our proposed scheme in detail. Our system is based on the concept of a consortium blockchain. Therefore, each agency can be seen as one member in the consortium blockchain network. Our scheme includes four phases: initialization, data sharing, shared data auditing, and user trace. In our system, we assume that there is a manager ($M_i, 1 < i \leqslant N$, where $N$ is the number of agencies) in each agency.

• **Initialization.**

(1) Parameter initialization. Input security parameter $\kappa > 1, k, l \in \mathbb{N}$. The system randomly chooses $\lambda_1, \lambda_2, \gamma_1, \gamma_2$, where $\lambda_1 > \kappa(\lambda_2 + k) + 2, \lambda_2 > 4l, \gamma_1 > \kappa(\gamma_2 + k) + 2, \gamma_2 > \lambda_2 + 2$. Define integral ranges $A = [2^{\lambda_1} - 2^{\lambda_2}, 2^{\lambda_1} + 2^{\lambda_2}]$, $B = [2^{\gamma_1} - 2^{\gamma_2}, 2^{\gamma_1} + 2^{\gamma_2}]$. Input security parameter $\kappa_1$, the bilinear Diffie-Hellman generator returns $\{q, g, \mathbb{G}_1, \mathbb{G}_2, \hat{e}\}$. $\mathbb{G}_1$ and $\mathbb{G}_2$ are the multiplicative group of large prime $q$, and $g_0$ is a generator of $\mathbb{G}_1$. The system chooses two hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_q$, $H_2 : \{0,1\}^* \to \mathbb{G}_1$. All the parameters above are published. The system generates public key pk and secret key sk by running KeyGen() (Algorithm 1) for each manager $M_j, j \in [1, N]$, where $N$ is the number of agencies.

**Algorithm 1** KeyGen()

**Require:** Prime numbers $p, q$ with $l$ bits;
**Ensure:** pk, sk;
1: Compute $P = 2p + 1, Q = 2q + 1$, let $n = PQ$;
2: Select random elements $g, h, a_1, a_2, g_1, g_2, \delta_1, \delta_2 \in_R QR(n)$, where $QR(n)$ denotes the quadratic residue of group $\mathbb{Z}_q^*$;
3: Select random $X_j \in \mathbb{Z}_q^*$, and compute $Y_j = g^{X_j}, j \in [1, N]$;
4: Let pk $= \{Y_j, g, h, a_1, a_2, g_1, g_2, \delta_1, \delta_2, n\}$, sk $= \{X_j, j \in [1, N]\}$;
5: **return** pk, sk;

Then, the system sends pk, sk to each manager.

(2) User registration. User $U_i$ registers with manager $M_i$ of the agency that he trusts and uploads data to. $U_i$ randomly chooses secret values $x_i' \in [0, 2^{\lambda_2}]$ and $r_i' \in [0, n]$ and computes $C_1 = g^{x_i'} g^{r_i'}$. Then, $U_i$ sends $C_1$ to manager $M_i$. $M_i$ checks if $C_1 \in QR(n)$. If the validation is successful, $M_i$ randomly chooses two values $\alpha_i, \beta_i \in [0, 2^{\lambda_2}]$ and sends them to user $U_i$ who computes $x_i = 2^{\lambda_1} + (\alpha_i x_i' + \beta_i \bmod 2^{\lambda_2})$ and $C_2 = a_1^{x_i}$. Then, the user sends $C_2$ to manager $M_i$. $M_i$ checks if $C_2 \in QR(n)$. If the validation is successful, $M_i$ randomly chooses $e_i \in B$, computes $A_i = (C_2^{a_2})^{\frac{1}{e_i}}$, and sends $(A_i, e_i)$ to user $U_i$ as his member certificate. After receiving $A_i, e_i$, user $U_i$ verifies that $a_1^{x_i a_2} = A_i^{e_i}$. If the validation is successful, the user sets $\{A_i, e_i\}$ as his signed key.

• **Data sharing.**

(1) Upload. In our construction, each shared data file generates a Merkle hash tree. When a user wants to upload the shared data, he should perform the following steps:

**Algorithm 2** GenSign()

**Require:** Secret key $\{A_i, e_i\}$, system parameters $\{Y_j, g, h, a_1, a_2, g_1, g_2, \delta_1, \delta_2, n\}$, and message $M$;
**Ensure:** Signature $\sigma$ on message $M$;
1: Select random number $r \in \{0,1\}^{2l}$, and compute the following values:
$T_1 = A_i Y_j^r$, $T_2 = g^r$, $T_3 = g^{e_i} h^r$;
2: Select random numbers:
$r_1 \in \pm\{0,1\}^{\kappa(\gamma_1+k)}$;
$r_2 \in \pm\{0,1\}^{\kappa(\lambda_2+k)}$;
$r_3 \in \pm\{0,1\}^{\kappa(\gamma_1+2l+k+1)}$;
$r_4 \in \pm\{0,1\}^{\kappa(2l+k)}$;
then, compute the following values:
$d_1 = T_1^{r_1}/(a_1^{r_2} Y_j^{r_3})$;
$d_2 = T_2^{r_1}/(g^{r_3})$;
$d_3 = g^{r_4}$;
$d_4 = g^{r_1} h^{r_4}$;
3: Generate a hash value
$c = H_1(g\|h\|Y\|a_1\|a_2\|T_1\|T_2\|T_3\|d_1\|d_2\|d_3\|d_4\|M)$;
4: Compute the following values:
$s_1 = r_1 - c(e_i - 2^{\gamma_1})$;
$s_2 = r_2 - c(x_i - 2^{\lambda_2})$;
$s_3 = r_3 - cre_i$;
$s_4 = r_4 - cr$;
5: Generate the signature:
$\sigma = (c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$;
6: **return** $\sigma$.

- The file denoted by $F$ is divided into $L$ blocks $f_1, f_2, \ldots, f_L$. The user uses hash values $H(f_k)$ as the leaf of the tree to build a Merkle hash tree [32]. Let $H_f$ denote $\{H(f_1), H(f_2), \ldots, H(f_L)\}$. Next, run GenSign() (Algorithm 2) to generate group signature $\sigma$ on the hash of the Merkle root $H(R)$. Let $\nu = f_{\mathrm{ID}}\|L$ be the tag for file $F$ and $f_{\mathrm{ID}}$ is the file ID. Then, the user sends shared data $D = (F, H_f, H(R), \sigma, \nu)$ to the manager.

- The manager verifies the validity of $D$. First, he runs Ver() (Algorithm 3) to verify the user identity. Then, he uses $(F, H_f)$ to rebuild the Merkle hash tree for the file and generates $H(R')$. Finally, the manager verifies whether $H(R) = H(R')$. After a successful verification, the manager stores the entire Merkle hash tree in the cloud server.

(2) Block generation. The manager builds a transaction that contains authentication information for the shared file. The authentication information consists of $(H(R), \sigma, L, f_{\mathrm{ID}}, \Omega)$. $\Omega$ denotes the information for the nodes to verify Merkle root $H(R)$. Then, the manager sends the transaction to the blockchain network. The nodes in the blockchain receive the transaction and verify the authentication information according to the pre-algorithm as follows:

- Merkle root verification. The blockchain nodes can verify $H(R)$ by using $\Omega$ to generate the hash of the Merkle root, $H(R')$, and check if $H(R) = H(R')$.

- Signature verification. The nodes can run the Ver() algorithm to verify signature $\sigma$ on $H(R)$.

After successful verification, the nodes run the practical Byzantine fault tolerant (PBFT) consensus algorithm. Then, the blockchain synchronizes this authentication information for the shared file. Finally, a block is produced in the blockchain. The structure of the block is shown in Figure 2. Each block contains four parts. The first is the hash value of the previous block, denoted by Prehash. The second part comprises the shared file ID, denoted by $f_{\mathrm{ID}}$, timestamp $t$, Merkle root of the file $H(R)$, and number $N$ of the file blocks corresponding to the shared file. The third part stores signature $\sigma$. The forth component contains a random number nonce. All the data in the block are public to the members of the system. All the users can browse the ledgers and read the information in the entire blockchain. Each manager in each agency preserves a pointer list, which can be checked by the data IDs. A pointer directly

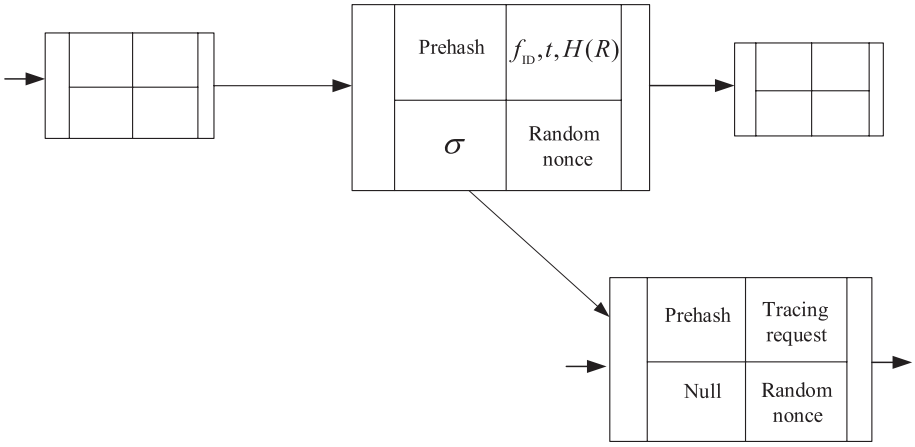


**Figure 2** The structure of the blocks.

**Algorithm 3** Ver()

**Require:** System parameters $\{Y_j, g, h, a_1, a_2, g_1, g_2, \delta_1, \delta_2, n\}$ and message $M$ with signature $\sigma$.
**Ensure:** True or False;
1: Compute the following values:
$d'_1 = a_1^c T_1^{s_1 - c2^{\gamma_1}} / a_1^{s_2 - c2^{\gamma_1}} Y_j^{s_3}$;
$d'_2 = T_2^{s_1 - c2^{\gamma_1}} / g^{s_3}$;
$d'_3 = T_2^c g^{s_4}$;
$d'_4 = T_3^c g^{s_1 - c2^{\gamma_1}} h^{s_4}$;
2: Generate a hash value
$c' = H_1(g \| h \| Y_j \| a_1 \| a_2 \| T_1 \| T_2 \| T_3 \| d'_1 \| d'_2 \| d'_3 \| d'_4 \| M)$;
3: **if** $c = c'$ **then**
4: return 'T';
5: **else**
6: return 'F';
7: **end if**
8: **return** $\sigma$;

points to the specific block corresponding to a file. Each manager publishes this list to all the users in the same group, and then the users can check this list.

- **Data auditing.**

Because the authentication information is recorded in a blockchain, all the users in the system can verify the shared data without the need for a trusted auditor. In our scheme, the users can obtain Merkle hash root $H(R)$ and signature $\sigma$ from the public blockchain and then obtain the auxiliary information by interaction with the cloud server. Finally, we use the technique of [29,33] for auditing. The data auditing process is depicted below.

(1) A user chooses integer $k$, where $k$ divides $N$ such that the leaves of the Merkle tree are divided into $k$ groups. Then, the user computes $s = H^k(R)$ and chooses one leaf number $\{t, 0 \leqslant t \leqslant k-1\}$ from each group and computes $k_t = P(s, t)$, where $P$ is a cryptographic pseudo-random number generator.

(2) The cloud server sends the appropriate sibling information to the user. After obtaining the auxiliary information, the user can construct a path to the Merkle root, compute $H(R')$, and verify whether $H(R) = H(R')$.

(3) The user computes $s' = H^k(R')$ and $k'_t = P(s', t)$ and verifies whether $k'_t = k_t$ for each $t$. If the equation holds, the data are verified.

- **User trace.**

In our scheme, when a dispute occurs, the real identity of the data owner can be revealed by the manager in each agency. Here, the manager in each agency can be seen as a consortium blockchain member. The process of tracing the origin of an identity is as follows.

(1) A blockchain member (the manager in each agency) makes a traitor tracing request on the blockchain

network. The blockchain nodes run the PBFT consensus algorithm. When at least two-thirds of the nodes confirm that the request is valid, the blockchain performs consensus and synchronization and generates new blocks. The current block is the proof of traitor tracing. Then, the member sends the proof and disputable data to the manager.

(2) The group manager computes $A_i = T_1/T_2^{X_j}$ and obtains the identity of signature $\sigma$.

In our proposed scheme, all the agencies work together to construct the consortium blockchain network. Each agency represents a consortium blockchain member, and each member operates a node. The nodes participating in the blockchain are selected in advance, and good network connections usually exist between these nodes. Before a new block is formed, the nodes need to agree on the transactions so that a distributed consensus mechanism is needed in the blockchain network. In our consortium blockchain network, we adopt PBFT as the consensus algorithm. In this algorithm, a "leader" node responsible for the generation of a new block should be selected in each round according to defined rules. The generation process involves three phases: pre-preparation, preparation, and commit. In each phase, a node can participate in the next phase if it has received votes from more than two-thirds of all the nodes. PBFT is based on the assumption that fewer than 1/3 of the nodes are faulty ($f$), where $f$ is the probability. Therefore, the whole system requires at least $3f + 1$ nodes.

## 4 Security analysis

In this section, we present a brief security analysis of our scheme.

**Theorem 1.** The proposed scheme satisfies the property of completeness.

*Proof.* In our data sharing structure, the actual data are stored in the cloud server instead of in the blockchain. Only the verification information is stored in the blockchain, which is a public ledger that can be read by each user in the system. Thus, the data are invisible to the blockchain nodes. Contrastingly, we also use group signature techniques to achieve anonymous information exchange in the public blockchain. Thus, the identity privacy of the users is protected. In our scheme, the sharing data are not required to be encrypted for storing in the storage server. To ensure data security in some specific scenarios, we adopt the symmetric balanced incomplete block design technique, which was proposed by Shen et al. [10], to generate a session key. The shared data are encrypted in advance by the common conference key, thereby making the data invisible to illegal users.

**Theorem 2.** The proposed scheme satisfies the property of anonymity.

*Proof.* In our scheme, the verification information of the data and group signature of the user are publicly available on the blockchain. Even if an attacker obtains the group signature, it is difficult for him to reveal the identity of the signer, $A_i$. According to the theorem in [34], the underlying interactive protocol is statistically zero-knowledge. Therefore, useful information of $(c, s_1, s_2, s_3, s_4)$ cannot be obtained from group signature $\sigma = (c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$. To determine which group member owns certificate $(A_i, e_i)$, one should check whether the three discrete logarithms, $\log_y(T_1/A_i)$, $\log_g(T_2)$, and $\log_g(T_3/g^{e_i})$ are equal. However, not anyone can perform this process; otherwise, the decisional Diffie-Hellman assumption will be violated. Hence, anonymity is guaranteed.

**Theorem 3.** The proposed scheme satisfies the property of traceability.

*Proof.* The manager of each group can obtain $A_i$ by using the value $X_i$. Then, he can reveal the real identity of the signer by checking the user list maintained by himself. When a dispute occurs, the manager can compute $A_i = T_1/T_2^{X_i}$ and obtain the identity of the signer.

**Theorem 4.** The proposed scheme satisfies the property of public auditability.

*Proof.* First, the verification information of the shared data is stored in a blockchain, which is a public ledger that can be read by each user in the system. All the users can validate their data without requiring a trusted third party. For traditional third-party auditing, the verifiers usually validate some randomly selected data blocks. However, the third party may collect personal information from the auditing data, and the data security is exposed to strong threats. The auditor may analyze some blocks that are not

**Table 1** Comparison of the schemes in terms of the verification process

| | Scheme [15] | Scheme [18] | Scheme [29] | Scheme [23] | Our scheme |
|---|---|---|---|---|---|
| Communication | $O(1)$ | $tO(\log L)$ | $tO(\log L)$ | $tO(\log L)$ | $tO(\log L)$ |
| Server computation | $O(t)$ | $tO(\log L)$ | $tO(\log L)$ | $tO(\log L)$ | $tO(\log L)$ |
| Verifier computation | $O(t)$ | $tO(\log L)$ | $tO(\log L)$ | $tO(\log L)$ | $tO(\log L)$ |

**Table 2** Functionality comparison of the schemes

| | Scheme [15] | Scheme [18] | Scheme [29] | Scheme [23] | Our scheme |
|---|---|---|---|---|---|
| Anonymity | No | No | No | Yes | Yes |
| Traceability | No | No | No | Yes | Yes |
| Number of groups | One | One | One | One | Multiple |
| Public auditing | No | Yes | Yes | Yes | Yes |
| Non-frameability | No | No | No | No | Yes |
| Third-party auditor | Yes | Yes | No | Yes | No |

selected for auditing and then collude with the cloud server. The data can be subject to tampering, which may not be detected. In our structure, the verification information contains the Merkle root, which is constructed from the entire data, so that even a single bit change is easily detected. Moreover, the Merkle root is stored in the blockchain, which is tamper-resistant. Once data have been written into the blockchain, they are protected from tampering.

In the following, we present a brief analysis of the tamper-resistance of the blockchain. If attacker $A$ wants to tamper with the data (i.e., the verification information) stored in the blockchain, then $A$ needs to control at least 51% of the computational power of the total blockchain network. Assume that there are $n$ nodes in the blockchain network. The probability that attacker $A$ controls one node in the blockchain is a non-negligible value, $\epsilon$. We can construct a random oracle (RO) that can provide queries for verification for the hash function. Attacker $A$ is allowed to make tq queries in each round, where $t$ is the number of nodes he controls. Assume that the height of the current block is $s$. If the attacker wants to modify $t$ blocks, then he should make $(t-s)(n/2+1)q$ queries. Probability of success $P$ satisfies $P > 1 \setminus (\epsilon^{n/2+1}(t-s) \setminus (n/2+1)q)$. When $n$ is large, the probability is non-negligible.

**Theorem 5.** The proposed scheme satisfies the property of non-frameability.

*Proof.* In our scheme, when a dispute occurs, the manager should make a traitor tracing request on the blockchain network before revealing the identity of the user. The manager can do this only when at least two-thirds of the nodes confirm that the request is valid. Thus, the tracking authority of the manager is decreased, so the hidden danger of power centralization is eliminated. Therefore, our proposed scheme satisfies the property of non-frameability, and fairness can be guaranteed in the process of tracking the true identity of a malicious user, i.e., honest users cannot be framed and dishonest users cannot be shielded.

## 5 Performance analysis

In this section, we first give a simplified comparison in Tables 1 and 2 [15,18,23,29] , and then we describe the experimental evaluation of our proposed scheme.

### 5.1 Comparison

Our data sharing scheme is based on the concept of a blockchain. The public verification information used for auditing is recorded in a tamper-resistant database, which is maintained by all the participants using the consortium blockchain technology. The users in the system can easily verify the integrity of the shared data without interacting with a third party. Moreover, we consider the case where multiple groups work together to establish a sharing database. Therefore, we compare only the communication

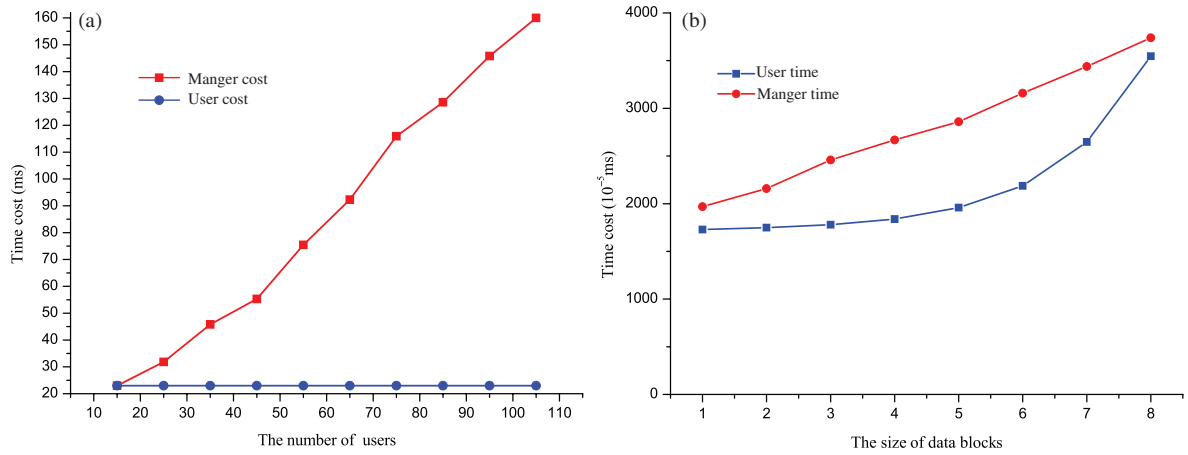


**Figure 3** (Color online) (a) Time cost of a user and manager in initialization; (b) time cost of a user and manager in data sharing.

overhead and computation costs for a single file in the auditing process. In our scheme, the verifier can be any user of the system.

Table 1 compares our scheme and the existing schemes. We consider a file that is divided into $L$ file blocks. $t$ denotes the number of sampled file blocks to be verified when verifying a file. Table 1 shows that the overheads of the communication and computation in our scheme are the same as those of the existing schemes. Table 2 compares the functionality of our scheme and the other schemes. Our scheme achieves data sharing functionality between multiple groups while simultaneously demonstrating the property of non-frameability, which is not considered in the existing schemes. Moreover, there is no extra overhead in the auditing process.

### 5.2 Performance evaluation

In this section, we provide experimental simulations for only the main time cost in each phase. The evaluation was implemented on the Amazon cloud server with a Dual Core i5 CPU and 8.0 GB RAM. The system used is Ubuntu 16.04. We use Python to realize our simulations. Group signature and verification algorithms are realized in Python. In the process of implementation, the key and parameter are 1024-bit-long.

To eliminate the need for a third-party auditor, we store the verification information in a blockchain. The generation of each block is implemented in the framework of Truffle. The language we use to code smart contracts is solidity. We use solidity in combination with TestRPC to exploit smart contracts. We perform the experiments by creating a private chain to simulate the operation of a smart contract and blockchain.

**Computation of initialization.** In the phase of initialization, the group manager must perform a $2n$-modular exponentiation, where $n$ is the number of users. In the phase of user registration, each user only perform two-modular exponentiation. Note that the overhead of the users is independent of the number of users in the system. Figure 3 shows the time cost of the user and manager during initialization. The result shows that our scheme is feasible in practice.

**Computation of data sharing.** In our scheme, the users need to build a Merkel hash tree for the shared data. Then they can obtain the Merkle root. The total time cost of the users is divided into two parts: the construction of the Merkel tree and generation of the signature on the Merkle root. Furthermore, the time for building the Merkel hash tree increases with the size of the file. It takes approximately 0.018 ms to create a Merkel tree when the size of a data file reaches 128 M. We only test the situation when the data file is small in size. Continuing the data size scaling, the time cost will increase. However, this part of the operation can be conducted off-line. The group manager should verify the Merkle root and signature. Therefore, the main overhead of the manager is a single-Merkle root verification and single-signature verification. Figure 3 shows the time cost of a user and group manager.

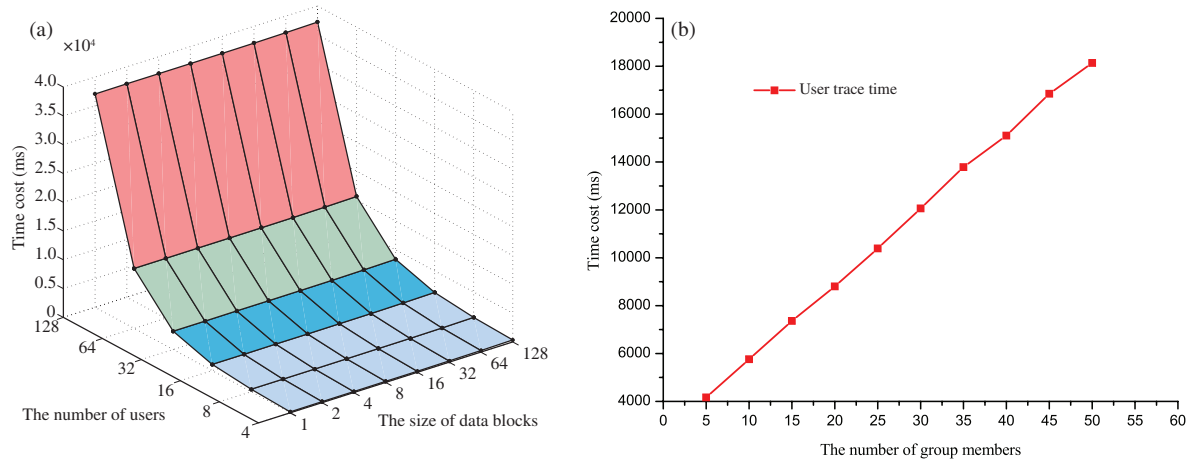


**Figure 4** (Color online) (a) The impact of data blocks size on the user audit overhead; (b) the impact of group members size on the user trace overhead.

The size of the data block is fixed to 128 M in our simulation.

To eliminate a third-party auditor, we store the verification information in a blockchain. In the phase of Data sharing, all the group managers should work together to build the blockchain. In our simulation, we experiment by creating a private chain to simulate the operation of a smart contract and blockchain. The system requires to perform communication thrice, a single smart contract, a single group signature verification, and a single Merkle hash root verification.

**Computation of shared data auditing.** The audit process of the shared data is implemented by any user in the system. The audit computational overhead consists of two parts: the computing overhead of the exponential power of the Merkel hash root and reconstruction of the Merkel hash tree. Figure 4(a) shows the impact of the data block size on the user audit overhead. In our experiment, we test the time cost when the number of group members increases from four to 128. When the size of a data block is large, the time cost increases continuously.

**Computation of traceability.** To trace the identity of a malicious user, the group manager publishes a transaction and waits for the generation of a new block. Then, he traces the real identity. In the simulation process, we only test the time overhead of the manager when there are six nodes in the blockchain. As shown in Figure 4(b), the time cost in the tracing process is low. It takes only 0.5 s for a group manager to trace the identity of a user. Furthermore, the tracking time of the group managers is not only related to the number of nodes in the blockchain but also to the size of the file block. When the number of nodes is large, the group managers use more time.

## 6 Conclusion

In this paper, we present an anonymous and traceable data sharing scheme for multiple groups. Based on the blockchain technique, the proposed scheme can realize secure data sharing reliably without requiring a trusted auditor. To remove the requirement of a trusted auditor, we store the public verification information for auditing in a tamper-resistant database that is maintained by all the participants using the consortium blockchain technology. Additionally, we adopt a group signature technique to achieve anonymity and traceability of the user identities. When a dispute occurs, the group manager can reveal the identity of the data owner. Moreover, fairness can be guaranteed in the process of solving the dispute.

**Acknowledgements** This work was supported by National Cryptography Development Fund (Grant No. MMJJ20180110).

**References**

1 Wang C, Wang Q, Ren K, et al. Privacy-preserving public auditing for data storage security in cloud computing. In: Proceedings of INFOCOM 2010, San Diego, 2010. 1–9

2 Chen X F, Li J, Ma J F, et al. New algorithms for secure outsourcing of modular exponentiations. IEEE Trans Parallel Distrib Syst, 2014, 25: 2386–2396

3 Liu X F, Zhang Y Q, Wang B Y, et al. Mona: secure multi-owner data sharing for dynamic groups in the cloud. IEEE Trans Parallel Distrib Syst, 2013, 24: 1182–1191
4 Wang C, Chow S S M, Wang Q, et al. Privacy-preserving public auditing for secure cloud storage. IEEE Trans Comput, 2013, 62: 362–375
5 Chen X F, Huang X Y, Li J, et al. New algorithms for secure outsourcing of large-scale systems of linear equations. IEEE Trans Inform Forensic Secur, 2015, 10: 69–78
6 Zhang X Y, Jiang T, Li K C, et al. New publicly verifiable computation for batch matrix multiplication. Inf Sci, 2019, 479: 664–678
7 Wang J F, Chen X F, Huang X Y, et al. Verifiable auditing for outsourced database in cloud computing. IEEE Trans Comput, 2015, 64: 3293–3303
8 Chen X F, Li J, Weng J F, et al. Verifiable computation over large database with incremental updates. IEEE Trans Comput, 2016, 65: 3184–3195
9 Zhang Z W, Chen X F, Ma J F, et al. SLDS: secure and location-sensitive data sharing scheme for cloud-assisted cyber-physical systems. Future Gener Comput Syst, 2018. doi: 10.1016/j.future.2018.01.025
10 Shen J, Zhou T Q, Chen X F, et al. Anonymous and traceable group data sharing in cloud computing. IEEE Trans Inform Forensic Secur, 2018, 13: 912–925
11 Chen X F, Li J, Huang X Y, et al. New publicly verifiable databases with efficient updates. IEEE Trans Dependable Secure Comput, 2015, 12: 546–556
12 Zhang Z W, Chen X F, Li J, et al. HVDB: a hierarchical verifiable database scheme with scalable updates. J Ambient Intell Human Comput, 2018, 53: 1–13
13 Yves D, Jean-Jacque Q, Ayda S. Remote integrity checking. In: Integrity and internal control in information systems VI. Boston: Springer, 2004. 1–11
14 Gazzoni Filho D L, Barreto P S L M. Demonstrating data possession and uncheatable data transfer. IACR Cryptol ePrint Archive, 2006, 2006: 150
15 Ateniese G, Burns R, Curtmola R, et al. Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, 2007. 598–609
16 Yang K, Jia X H. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans Parallel Distrib Syst, 2013, 24: 1717–1726
17 Yuan J, Yu S. Efficient public integrity checking for cloud data sharing with multi-user modification. In: Proceedings of INFOCOM 2014, Toronto, 2014. 2121–2129
18 Erway C C, Küpçü A, Papamanthou C, et al. Dynamic provable data possession. ACM Trans Inf Syst Secur, 2015, 17: 1–29
19 Yuan H R, Chen X F, Jiang T, et al. DedupDUM: secure and scalable data deduplication with dynamic user management. Inf Sci, 2018, 456: 159–173
20 Wang J F, Chen X F, Li J, et al. Towards achieving flexible and verifiable search for outsourced database in cloud computing. Future Gener Comput Syst, 2017, 67: 266–275
21 Zhang X Y, Chen X F, Wang J F, et al. Verifiable privacy-preserving single-layer perceptron training scheme in cloud computing. Soft Comput, 2018, 22: 7719–7732
22 Ma X, Zhang F G, Chen X F, et al. Privacy preserving multi-party computation delegation for deep learning in cloud computing. Inf Sci, 2018, 459: 103–116
23 Wang Q, Wang C, Ren K, et al. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans Parallel Distrib Syst, 2011, 22: 847–859
24 Huang H, Chen X F, Wu Q H, et al. Bitcoin-based fair payments for outsourcing computations of fog devices. Future Gener Comput Syst, 2018, 78: 850–858
25 Huang H, Li K C, Chen X F. Blockchain-based fair three-party contract signing protocol for fog computing. Concurr Computat Pract Exper, 2018, 28: e4469
26 Yang C S, Chen X F, Xiang Y. Blockchain-based publicly verifiable data deletion scheme for cloud storage. J Netw Comput Appl, 2018, 103: 185–193
27 Liang J, Han W L, Guo Z Q, et al. DESC: enabling secure data exchange based on smart contracts. Sci China Inf Sci, 2018, 61: 049102
28 Gaetani E, Aniello L, Baldoni R, et al. Blockchain-based database to ensure data integrity in cloud computing environments. In: Proceedings of Italian Conference on Cybersecurity, Venice, 2017. 1816: 146–155
29 Ghoshal S, Paul G. Exploiting block-chain data structure for auditorless auditing on cloud data. In: Proceedings of International Conference on Information Systems Security, Jaipur, 2016. 10063: 359–371
30 Chaum D, van Heyst E. Group signatures. In: Proceedings of Workshop on the Theory and Application of of Cryptographic Techniques, Berlin, 1991. 257–265
31 Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf
32 Merkle R. Secrecy, authentication, and public key systems. Dissertation for Ph.D. Degree. Stanford: Stanford University, 1979
33 Coelho F. An (almost) constant-effort solution-verification proof-of-work protocol based on merkle trees. In: Proceedings of International Conference on Cryptology in Africa, Casablanca, 2008. 80–93
34 Ateniese G, Camenisch J, Joye M, et al. A practical and provably secure coalition-resistant group signature scheme. In: Proceedings of Annual International Cryptology Conference, Berlin, 2000. 255–270