Liu, Gao; Dong, Huidong; Yan, Zheng; Zhou, Xiaokang; Shimizu, Shohei

**B4SDC: A Blockchain System for Security Data Collection in MANETs**

*Published in:*
IEEE Transactions on Big Data

*DOI:*
10.1109/TBDATA.2020.2981438

E-pub ahead of print: 01/01/2020

*Document Version*
Peer reviewed version

*Please cite the original version:*
B4SDC: A Blockchain System for Security Data Collection in MANETs

Gao Liu, Huidong Dong, Zheng Yan, Senior Member, IEEE, Xiaokang Zhou, Shohei Shimizu

Abstract—Security-related data collection is an essential part for attack detection and security measurement in Mobile Ad Hoc Networks (MANETs). A detection node (i.e., collector) should discover available routes to a collection node for data collection and collect security-related data during route discovery for determining reliable routes. However, few studies provide incentives for security-related data collection in MANETs. In this paper, we propose B4SDC, a blockchain system for security-related data collection in MANETs. Through controlling the scale of Route REQuest (RREQ) forwarding in route discovery, the collector can constrain its payment and simultaneously make each forwarder of control information (namely RREQs and Route REPlies, in short RREP) obtain rewards as much as possible to ensure fairness. At the same time, B4SDC avoids collusion attacks with cooperative receipt reporting, and spoofing attacks by adopting a secure digital signature. Based on a novel Proof-of-Stake consensus mechanism by accumulating stakes through message forwarding, B4SDC not only provides incentives for all participating nodes, but also avoids forking and ensures high efficiency and real decentralization. We analyze B4SDC in terms of incentives and security, and evaluate its performance through simulations. The thorough analysis and experimental results show the efficacy and effectiveness of B4SDC.

Index Terms—MANETs, security-related data collection, incentive mechanism, blockchain.

1 INTRODUCTION

MANET suffers from different attacks due to self-organization [1]. In order to provide a secure and high-quality networking service, security-related data collection becomes essential for network attack detection and security measurement. Security-related data, in short security data, are the data that can be used to discover network threats and measure its security. In data collection, after receiving the request of a collector or detection node, collection nodes send it sensed security data [2]. Since there is no fixed infrastructure in MANETs, nodes cooperate to forward the request and security data. In order to ensure efficient collection, a collector should discover available routes to collection nodes through route discovery, so that the request and security data can be transmitted via these routes. However, the route discovery suffers from various attacks, e.g., wormhole and rushing attacks [1]. Existing detection mechanisms can help a collector to detect these attacks and select reliable routes [13], but the collector should analyze security data provided by forwarders of control information for making decisions, e.g., the timestamps of receiving and sending a packet, the location of forwarders. Unfortunately, few studies provide incentives for security data collection in MANETs. Forwarders might not be willing to sense and provide security data due to extra overload, selfishness, etc., so that the collector could not collect sufficient security data for detection. As a result, the accuracy of threat detection and security measurement cannot be ensured.

Current methods to provide incentives mainly make use of reputation and micropayment systems [3]. However, the present studies in this field are still facing a number of issues to be applied into security data collection in MANETs. First, these mechanisms do not consider spoofing attacks that an attacker launches for maximizing its profits due to no identity management. Second, reputation systems might not resist collusion attacks that selfish nodes raise for improving their reputation, and do not specify the type of incentive (e.g., what profit can be brought by high reputation.) Furthermore, a highly reputable node might suddenly have malicious behaviors. Third, micropayment systems allow a collector to pay to collection nodes and forwarders for their cooperation and contributions, but most of them require a trusted third party to manage debiting and crediting accounts. This kind of design is obviously infeasible for MANETs since such a trusted third party is hard to be deployed. Fourth, in some micropayment systems for route discovery, a source node pays control information forwarders, but cannot constrain its payment, which is caused by an uncontrolled RREQ forwarding scale [3]. In addition, some existing systems allow the source node to only pay the forwarders in discovered routes, which is unfair to the nodes that have participated in route discovery but are not in the routes [5].

In order to solve the above issues, a distributed incentive system for security data collection is highly expected, which should resist spoofing, collusion and excessive forwarding, and feature fairness, and should not rely on a trusted third party. We found that blockchain [6], [7] is a candidate technology to help achieving the above goals due to its advantages, e.g., transparency, immutability, and
self-organization. In a blockchain system, a miner collects transactions, generates a block and provides it to other miners with a proof of work (e.g., the proof of computing and storage) to gain the majority of acceptance, which is called consensus. In general, the incentive for miners is provided in a form of digital tokens, e.g., bitcoin [8]. At the same time, the transaction can help incenting security data collection.

However, the blockchain itself is still facing many technical challenges, namely forking, low efficiency and a trend of centralization. Based on the current literature review [9], [10], consensus mechanisms mainly include Proof of Work (PoW), Byzantine Faulty Tolerant (BFT), Proof of Stake (PoS), Proof of Useful Work (PoUW) and Trees and Directed Acyclic Graphs (DAGs). PoW and PoUW take the risk of temporary forking due to network latency. PoW wastes a lot of resources since it is a meaningless task. Its transaction confirmation time is long, thus negatively impacting its throughput. PoW and PoUW take the risk of centralization due to the outsourceability of tasks. Because PoS consumes almost no resources, a miner might create two blocks to cause forking. Many BFT based consensus mechanisms focus on scalability, but they provide no incentives for miners. Trees and DAGs can replace the chain structure of blockchain for ensuring a high throughput and avoiding double spending, but some trees and DAGs based consensus mechanisms [11], [12] also employ PoW, thus suffering from the same problems of PoW, namely forking, low efficiency and the risk of centralization.

In this paper, we propose B4SDC, a blockchain system that provides incentives for security data collection in a distributed way in MANETs. Through controlling the scale of RREQ forwarding in route discovery, a collector can constrain its payment and simultaneously make each forwarder of control information obtain rewards as much as possible to ensure fairness. At the same time, B4SDC avoids collusion attacks with cooperative receipt reporting, and spoofing attacks by adopting a secure digital signature. Based on a novel Proof-of-Stake consensus mechanism by accumulating stakes through message forwarding, B4SDC not only provides incentives for all participating nodes, but also avoids forking and ensures high efficiency and real decentralization at the same time. With the above ways, B4SDC successfully avoids collusion and spoofing attacks, allows collectors to control their maximum payments, and ensures fairness for collection participants as much as possible. It also solves current blockchain systems’ main problems, namely forking, low efficiency and centralization. Specifically, the contributions of this paper can be summarized as follows.

1. B4SDC provides incentives for collection participants. It encourages nodes to forward control information that includes security data in route discovery. After routes are discovered, the nodes in selected routes are incented to forward the request of a collector and the security data of collection nodes.

2. B4SDC removes the need for a trusted third party in many micropayment systems by adopting blockchain. It adopts a secure digital signature for signing sent messages, thus avoiding spoofing attacks. At the same time, B4SDC allows collection participants to cooperatively report their received receipts to miners for gaining rewards, thus resisting collusion attacks and ensuring fairness for all collection participants as much as possible. In addition, a collector can constrain the scale of RREQ forwarding in route discovery, thus it can balance its budget.

(3) B4SDC provides a novel consensus mechanism. Block creation is proposed to ensure the distribution and efficiency of blockchain by avoiding the simultaneous generation of many valid blocks and reducing communication burdens. Single block winner selection is performed to make B4SDC free from forking when multiple valid blocks are created at the same time.

The rest of this paper is organized as follows. Section 2 briefly overviews background and related work. Section 3 and Section 4 present problem statements and the design of B4SDC, respectively, followed by analysis and performance evaluation in Section 5. Section 6 concludes the paper.

2 Background and Related Work

2.1 An Overview of Route Discovery

Mainstream routing protocols adopt route discovery [13] for data transmission, such as DSR and AODV protocols [29], [30]. In the route discovery, a source node floods a RREQ including its and destination node’s addresses. When a node receives the RREQ and it is not the destination node and has no routes to the destination node, it adds its address into the RREQ and then forwards this RREQ. If the destination node or an intermediate node having routes to the destination node receives the RREQ, it creates a RREP including a whole route path and sends the RREP to the source node in the reverse of the path.

2.2 Incentive Systems

Current incentive systems mainly include reputation and micropayment systems. Most of them rarely take spoofing attacks [32], [33] into consideration, since no identity management is adopted for nodes.

Reputation Systems: Nodes monitor traffic from their neighbors for determining whether these neighbors forward packets. If a node finds its neighbor does not forward packets, it considers the neighbor uncooperative and broadcasts an uncooperative reputation into the network [3], [4]. However, reputation systems do not specify the type of incentive, especially what profits a node can obtain with a high reputation. In addition, selfish nodes could collude to improve their reputation, and a highly reputable node could suddenly have malicious behaviors [23].

Micropayment Systems: Many micropayment systems allow nodes to forward packets for obtaining receipts. These nodes can send the obtained receipts to a third party for crediting and debiting accounts [3], [34]. However, it is not feasible to deploy this third party in MANETs, since MANETs have no fixed infrastructure and thus have no any management center. In addition, the party might be hacked or compromised. In some micropayment systems [3] for route discovery, a source node encourages the cooperation of control information forwarders for discovering routes to destination nodes, thus it pays these forwarders for incenting them. However, it cannot control the scale of RREQ
forwarding, which suggests it cannot limit the amount of payment. In some systems [5], when a route is discovered, the source node only pays control information forwarders in the route, but other nodes that have participated in route discovery but are not located in the route cannot obtain rewards. Thus, the fairness to all participants in route discovery is not guaranteed.

2.3 Blockchain Technology

PoW Based Consensus Mechanisms: Bitcoin employs PoW to make each node agree on some transactions. Due to temporary forking caused by network latency, users wait at least six blocks (i.e., expected 1 hour) for confirming a transaction according to the longest chain principle. It also wastes many resources for solving a puzzle. At the same time, bitcoin also faces the risk of centralization due to the outsourceability of mining tasks. Many consensus mechanisms that adopt PoW have the limitations of bitcoin, such as Permacoin [22] that repurposes the storage of bitcoin.

PoUW Based Consensus Mechanisms: PoUW is usually a non-interactive proof of meaningful computation and storage that can convince any miners. Many PoUW based consensus mechanisms only construct the proof, but fail to show how to achieve the consistency of blockchain. Therefore, they suffer from temporary forking due to network latency [22]. In addition, many PoUW consensus mechanisms allow to split a task into subtasks, which might result in the outsourceability of tasks [26]. As a consequence, they take the risk of centralization.

BFT Based Consensus Mechanisms: BFT agreement protocols allow a set of servers to replicate a service. In much work [24] on BFT, its performance and scalability are emphasized as challenges. However, the set of servers should be chosen and determined in advance, and these servers might become attack targets. If everyone has the right to be a server, Sybil attacks [25] are a potential risk. BFT based consensus mechanisms inherit the limitations of BFT. In addition, BFT based consensus mechanisms face the challenge of lack of incentives for motivating determined servers to be online and perform honestly [9].

PoS Based Consensus Mechanisms: In PoS, each user has its weight, which represents the probability that it is chosen as a leader to create the next block of blockchain. Because creating a block costs almost no resources, a malicious miner might create two blocks, thus causing forking. Meanwhile, adversaries can divide their credits among some users that might be chosen as leaders. Therefore, although a malicious leader is discovered, the penalty that the adversaries take is low. In some PoS based consensus mechanisms, a private key is required for signing the correct branch of blockchain in order to mitigate forking, but occasional forks have appeared [27]. PoS based consensus mechanisms also have the risk of centralization due to the outsourceability of tasks [28].

Trees and DAGs Based Consensus Mechanisms: In order to increase the throughput of bitcoin and solve conflicts (i.e., double spending), some consensus mechanisms [11], [12] were proposed that adopt a tree or DAG structure as the blockchain’s underlying ledger instead of chains. However, these consensus mechanisms rely on PoW, thus having its limitations.

3 Problem Statements

3.1 System Model

As shown in Fig. 1, B4SDC involves three types of entities: certificate and task issuer, collection participant and miner. The certificate and task issuer is responsible for issuing certificates and tasks, which can be a blockchain system for key management, PKI or identity management to ensure decentralization [14], [15], [16]. It does its work honestly for rewards since the published certificates and tasks should obtain the consensus of B4SDC blockchain.

In the receipt collection layer, collection participants involve collectors, forwarders and collection nodes. The collectors discover routes to collection nodes with route discovery, and select some reliable routes to publish requests and receive security data from collection nodes, since the route discovery allows the collectors to collect security data from each forwarder for detecting potential attacks and obtaining reliable routes. The forwarders help forwarding received messages including security data. Each collection node and forwarder can use received messages as receipts and share these receipts to miners. In this layer, each node is not fully trusted since it might forge receipts for rewards.

In the consensus layer, after collecting sufficient receipts and receiving a task, the miners create and publish blocks for consensus. The majority of miners accept one block as the next block of blockchain. It is hard to ensure each miner is trusted since the miner might be hacked or compromised.

3.2 Assumption

Suppose an adversary can control a small number of miners, but its control is limited. It is impossible that all the miners are attacked and functionally down. The remaining miners behave independently and are rational to maximize their gains when investing resources into mining. The time of all the miners is supposed to be synchronized, which can be achieved with the help of public GPS signals [17] or a public time blockchain [18]. Because time synchronization can be achieved in the level of nanoseconds. Although this assumption restricts B4SDC, it is justified compared to secure incentive provisions for security data collection. In addition, miners cannot forge timestamps, since miners in the network are monitored by their neighbors and nodes can employ some lightweight external methods (e.g., wormhole attack detection [17]) to detect this kind of misbehavior.

Each node holds its own unique private/public keys in Elliptic Eurve Cryptosystem (ECC) and uses the private key...
for signing messages. This can be achieved by allowing each node to register at the certificate and task issuer. Otherwise, potential attacks might disturb security data and receipt provision and detection mechanisms. Suppose that nodes do not attempt to share their private keys to other nodes, since such sharing suggests their tokens are shared.

4 B4SDC DESIGN

In this section, we describe the design of B4SDC, including the block structure and receipt collection. A novel consensus mechanism is introduced for the block creation and block winner selection with an incentive mechanism. We discuss how to balance the budget of collector for route discovery and security data item collection finally. TABLE 1 summarizes the notations used in this paper.

4.1 Block Structure

The block structure of B4SDC is shown in Fig. 2. In the block $K$’s header, $T$ is a task that the certificate and task issuer publishes. $h_{K-1}$ is the hash value of previous block header (i.e., the header of block $K-1$). $pk$ is the ECC public key of creator of block $K$, and $CE_{mr}$ is the certificate corresponding to $pk$ that is issued by the certificate and task issuer. $t$ represents the time when the block $K$ is created. In the block $K$’s body, $TX_{mr}$ is a transaction that a miner uses to consume the age of some tokens for competing for the block creator. $TX_{vr}$, $v = 0, \ldots, L$ is the receipt of node $N_v$ collected when it participates in route discovery and security data item collection. $TX_{ctv}$, $q = 0, \ldots, X$ and $TX_{ccp}$, $p = 0, \ldots, Y$ are fresh deposit messages for route discovery and security data item collection. In addition, $mt$ in the header represents the root of Merkle tree constructed with $TX_{mr}$, $TX_{vr}$, $v = 0, \ldots, L$, $TX_{ctv}$, $q = 0, \ldots, X$ and $TX_{ccp}$, $p = 0, \ldots, Y$.

4.2 Receipt Collection

Before the consensus, receipt collection should be performed. It consists of Node Registration, Receipt Generation, and Receipt Sharing.

1) Node Registration

Each node provides its public key to and obtain its certificate from the certificate and task issuer.

1. Each node $N_i$ generates a pair of private/public keys $(sk_i, pk_i)$ based on ECC, and uses $pk_i$ to register at the certificate and task issuer $N_{T1}$.

2. $N_{T1}$ equipped with a pair of public/private keys $(sk_{T1}, pk_{T1})$ generates the certificate $CE_i = \{pk_{T1}, \text{sig}_{sk_{T1}}(pk_i)\}$ with Elliptic Curve Digital Signature Algorithm (ECDSA) [35] and issues it to $N_i$.

2) Receipt Generation

A collector discovers routes to some collection node with route discovery, which involves security data collection for detection since potential attacks might exist in the route discovery. The collector adopts reliable discovered routes to send a request and collect security data from collection nodes. Each of collection participants (except the collector) considers its received messages as receipts.

Route Discovery

A collector discovers routes to a collection node by broadcasting a RREQ and receiving RREPs. Each participant of route discovery (except the collector) considers its received control messages as receipts.

1. Through neighbor discovery, each node $N_i$ records the public key list $NL_i$ of its neighbors, and periodically publishes $SNL_i = \{CE_i, pk_i, NL_i, t_i, \text{sig}_{sk_i}(NL_i, t_i)\}$.

2. Based on collection strategies, the collector $N_0$ attempts to discover routes to the selected collection node $N_d$ for security data collection. In order to avoid the collector’s unreliable payment (e.g., double spending), $N_0$ should send a blockchain (i.e., B4SDC blockchain, not $N_{T1}$) the deposit message $TX_d = \{CE_{i}, pk_{i}, \text{UID}, NU_0\}$ to deposit $NU_0$ tokens for the unique identity $UID$ of RREQ in advance. If $N_0$ makes sure that its message $TX_d$ is inserted into blockchain, it can use $NU_0$ tokens as the budget for route discovery. Then it determines appropriate Time To Live ($TTL$) and $k = \{k_h : h = 0, \ldots, T TL - 1\}$ due to its budget, which is discussed in Section 4.4 later on. $k_h$ is the number of neighbors that each node at hop $h$ selects as the receivers of RREQ.
3. $N_0$ randomly selects its $k_0$ neighbors $N_{00}, \ldots, N_{0k_0}$ as the receivers of its RREQ, whose public keys are $pk_{00}, \ldots, pk_{0k_0}$, respectively. Then it computes $BI = (CE_0, pk_0, UID, pk_d, TTL, k)$, and obtains its RREQ $Q_0 = \{BI, h = 0, RL, pk_{i1}, \ldots, pk_{i+k_i-1}, SD, t_i\}$ with the signature $\sigma_i = sig_{sk_i}(Q_0)$, where $pk_d$ is the public key of destination node $N_d$, $RL$ the route list, $SD$ the security data list including $N_i$’s security data, and $t_i$ the time of sending $Q_0$. If $|NL_0| < k_0$, $N_0$ selects its all neighbors as RREQ receivers.

4. After receiving $Q_i = \{BI, CE_i, pk_i, h, RL, pk_{i1}, \ldots, pk_{i+k_i-1}, SD, t_i\}$ and $\sigma_i = sig_{sk_i}(Q_i)$ from $N_i$ for the first time, $N_{i+1}$ can check that the collector does not make an unreliable payment by accessing the blockchain, its public key $pk_{i+k_i}$ belongs to $\{pk_{i1}, \ldots, pk_{i+k_i}\}$ and $h < TTL$, and then verifies $CE_0$ in $BI$, $CE_i$ and $\sigma_i$. If all verifications hold, $N_{i+1}$ saves $Q_i$ and $\sigma_i$ as receipts, then increases $h$ by 1, inserts its public key into $RL$, selects $k_{i+1}$ neighbors whose public keys are $pk_{i+k_i}, \ldots, pk_{i+k_{i+1}-1}$, adds its sensed security data into $SD$, and generates $Q_{i+1} = \{BI, CE_{i+1}, pk_{i+k_i}, h, RL, pk_{i+k_i+1}, \ldots, pk_{i+k_{i+1}}, SD, t_{i+1}\}$ and $\sigma_{i+1} = sig_{sk_{i+1}}(Q_{i+1})$. Finally, it broadcasts $Q_{i+1}$ and $\sigma_{i+1}$.

In order to prevent $N_i$ from selecting few receivers by deliberately inserting less than $k_i$ public keys in the RREQ for saving resources, each node $N_i$ can serve as an observer for monitoring the behaviors of its neighbor. If $N_i$ finds that the neighbor list $NL_i$ of its neighbor $N_j$ satisfies $|NL_j| \geq k_i$ and $N_i$ forwards $Q_i$ to less than $k_i$ neighbors, it can send the blockchain the report $TX_{vi} = \{CE_v, pk_v, RP_v = (SNL_v, Q_i, \sigma_i), SIG_v = sig_{sk_i}(RP_v)\}$, thus obtaining rewards by deducting $N_i$’s rewards. Therefore, nodes are inclined to provide security data and forward the RREQ.

5. When the destination node $N_d$ or the intermediate node $N_w$ having a route to the destination node receives the RREQ with a signature, it can check that the collector makes a reliable payment by accessing the blockchain, its public key belongs to the public key list in the RREQ and $h < TTL$. If all verifications hold, $N_d$ or $N_w$ saves the received RREQ and corresponding signature as receipts. Then $N_d$ inserts its sensed security data into $SD$, and generates the RREP $P_d = \{BI, CE_d, pk_d, RL, SD, sig_{sk_d}(BI, RL)\}$ with the signature $\delta_d = sig_{sk_d}(P_d)$. $N_w$ adds its sensed security data into $SD$, and creates the RREP $P_w = \{BI, CE_w, pk_w, RL, SD, sig_{sk_w}(BI, RL)\}$ with $\delta_w = sig_{sk_w}(P_w)$. Note that RREP’s $RL$ is a whole route path from $N_0$ to $N_d$, and $SD$ has all security data recorded in the received RREQ. Finally, $N_d$ or $N_w$ forwards its RREP with the signature in the reverse direction of route from $N_0$ to $N_d$.

6. When $N_i$ on route (derived from $RL$) receives a RREQ and a signature from $N_{i+1}$, namely $P_{i+1} = \{BI, CE_{i+1}, pk_{i+1}, RL, SD, sig_{sk_{i+1}}(BI, RL)\}$ or $P_{i+1} = \{BI, CE_{i+1}, pk_{i+1}, RL, SD, sig_{sk_{i+1}}(BI, RL)\}$ with $\delta_{i+1} = sig_{sk_{i+1}}(P_{i+1})$, it verifies the involved signatures. If all verifications hold, $N_i$ saves $P_{i+1}$ and $\delta_{i+1}$ as receipts, then inserts its sensed security data into $SD$, and generates and sends $P_i$ and $\delta_i$ to $N_{i-1}$ en route. When $N_0$ receives valid $P_i$ and $\delta_i$, the common neighbor $N_w$ of $N_0$ and $N_i$ can collect $P_i$ and $\delta_i$, and share $TX_x = \{CE_x, pk_x, RP_x = (SNL, SNL_1, P_1, \delta_1), SIG_x = sig_{sk_x}(RP_x)\}$ to the blockchain for rewards.

An example of route discovery is shown in Fig. 3. The collector $N_0$ sets $TTL = 2$, $k_0 = k_1 = 2$. $N_0$ selects two receivers $N_{i0}$ and $N_{i1}$, and broadcasts the RREQ $Q_0 = \{pk_{i0}, pk_{i1}\}$ with its signature $\sigma_0$, where $pk_{i0}$ and $pk_{i1}$ represent the public key of $N_{i0}$ and $N_{i1}$, respectively. $N_{i0}$ and $N_{i1}$ verify $Q_0 = \{pk_{i0}, pk_{i1}\}, \sigma_0$. If all verifications hold, $N_{i0}$ and $N_{i1}$ broadcast $\{Q_1, \{pk_{i2}, pk_{i3}\}, \delta_1\}$, and $\{Q_1, \{pk_{i4}, pk_{i5}\}, \delta_1\}$. At hop $h = 2$, $N_d$ and $N_2$, that has a route to $N_d$ receives RREPs, and verifies their received RREPs. If all verifications hold, $N_d$ generates the RREP with its signature, $\{\delta_2, \delta_2\}$, and $N_d$ creates $\{P_2, \delta_2\}$. $N_d$ and $N_2$ send their generated RREPs and corresponding signatures to $N_0$ through the reverse path. $N_0$ discovers two routes, $N_0 \rightarrow N_{i0} \rightarrow N_{i1} \rightarrow N_d$ and $N_0 \rightarrow N_{i1} \rightarrow N_d$. Each node (except $N_0$) stores its received messages as receipts, and shares them to the blockchain for rewards.

Security Data Item Collection
A collector sends a request to collection nodes, and then receives security data items from these nodes. Each participant of security data item collection (except the collector) regards its received messages as receipts.

1. When the collector $N_0$ has discovered routes to collection nodes, it can determine the reliable route $PA_{d0}$ to each collection node $N_d$ for security data collection (e.g., by adopting wormhole attack detection). These collection nodes and corresponding routes are denoted as $\{N_{d0}, \ldots, N_{d_{d0}}\}$ and $PA = \{PA_{d0}, \ldots, PA_{d_{d0}}\}$ respectively. The collector also deposits some tokens in advance in order to avoid unreliable payments. In detail, it sends the blockchain the deposit message $TX_{ce} = \{CE_0, pk_0, DI, NU_1\}$ to deposit $NU_1$ tokens, where $DI$ is the unique identity of request. If the collector knows $TX_{ce}$ has been added into the blockchain, it can start to generate and broadcast the request with the unique identity $DI$.

2. $N_0$ generates and broadcasts the request $DT_0 = \{CE_0, pk_0, DI, PA, sig_{sk_0}(DI, PA)\}$ with the signature $\varphi_0 = sig_{sk_0}(DT_0)$.

3. After receiving $DT_i = \{DT_0, CE_i, pk_i\}$ and $\varphi_i = sig_{sk_i}(DT_i)$ from the neighbor $N_i$, the node $N_{i+1}$ can check that the collector makes a reliable payment by accessing the blockchain, uses $DT_i$’s $PA$ to check it is
a legitimate receiver, and verifies involved signatures. If all verifications hold, $N_{i+1}$ considers $DT_i$ and $\varphi_i$ as receipts, then creates and broadcasts the new request $DT_{i+1} = \{DT_0, CE_{i+1}, pk_{i+1}\}$ with the signature $\varphi_{i+1} = \text{sig}_{sk_{i+1}}(DT_{i+1})$.

4. When the destination node $N_d$ receives the request from $N_0$, it checks the payment of collector is reliable by accessing blockchain, then checks it is a legitimate receiver and verifies the validity of request through involved signatures. If all verifications hold, $N_d$ considers the received request and signature as receipts, then creates the security data item $SDI_{d_j} = \{DT_0, CE_d, pk_d, SD_{d_i}, \text{sig}_{sk_d}(DT_0, SD_{d_i})\}$ with the signature $\phi_{d_j} = \text{sig}_{sk_d}(SDI_{d_j})$, and then forwards $SDI_{d_j}$ and $\phi_{d_j}$ to $N_0$ in the reverse of route from $N_0$ to $N_d$.

5. If $N_i$ receives $SDI_{i+1} = \{SDI_{d_j}, CE_{i+1}, pk_{i+1}\}$ and $\phi_{i+1} = \text{sig}_{sk_{i+1}}(SDI_{i+1})$, it verifies the validity of $SDI_{i+1}$ and $\phi_{i+1}$. If all verifications hold, $N_i$ saves $SDI_{i+1}$ and $\phi_{i+1}$ as receipts, then generates and forwards $SDI_i = \{SDI_{d_j}, CE_i, pk_i\}$ with $\phi_i = \text{sig}_{sk_i}(SDI_i)$ in the reverse of route from $N_0$ to $N_d$. As a result, $N_0$ can obtain the security data $SD_{d_i}$ from $N_d$. When $N_0$ receives valid $SDI_1$ and $\phi_1$, the common neighbor $N_0$ of $N_i$ and $N_1$ can collect $SDI_1$ and $\phi_1$, and send $TX_e = \{CE_i, pk_e, RP_e = (SNL_0, SNL_1, SDI_1, \phi_1), SIG_v = \text{sig}_{sk_e}(RP_v)\}$ to the blockchain for rewards.

We show an example of security data item collection in Fig. 4. $N_0$ has an idea about routes to $N_{d_0}$, $N_{d_2}$ in advance. After receiving a request that includes the routes and verifying its validity, $N_{d_0}$, $N_{d_2}$ and $N_{d_2}$ generate security data items with signatures, and send them to $N_0$ via the reversed path of the route. Each node (except $N_0$) stores received messages as receipts, and send them to the blockchain for rewards.

**Receipt Sharing**

1. Each node $N_v$ broadcasts $TX_e = \{CE_i, pk_e, RP_e, SIG_v\}$, where $RP_e = \{Q_i, \sigma_e\}$, $\{P_{v_i}, \delta_{v_i}\}$, $\{DT_{v_i}, \varphi_{v_i}\}$, $\{SDI_{v_i}, \phi_{v_i}\}$ or $\{SNL_0, SNL_1, \sigma_i\}$, and $SIG_v = \text{sig}_{sk_e}(RP_v)$ is $N_v$’s signature on $RP_v$.

2. Miners collect and verify $TX_e$ with new $UID$, $TX_{ec}$ with new $DI$ and $TX_{ec}$, since a miner will obtain rewards in the case that it successfully adds these messages into the blockchain.

**4.3 Consensus Mechanism**

We propose a novel consensus mechanism for B4SDC, which aims to solve three main technical challenges of blockchain, i.e., forking, low efficiency, and centralization. Based on the last block of blockchain, miners in different locations compete to create the next block, and only one of published valid blocks is accepted as the next block. The consensus mechanism consists of block creation, block winner selection and an incentive mechanism.

**Block Creation**

$T_{F1}$ publishes the task $T = \{pk_{F1}, TID, n_b, n_{othr}, n_{thr}, TAG, \theta, \text{sig}_{sk_{F1}}(TID, n_b, n_{othr}, n_{thr}, TAG, \theta)\}$ periodically, which triggers the block creation. $TID$ is the unique identity of $T$. $n_b$ is the number of the latest referenced blocks of the blockchain. $n_{othr}$ represents the threshold of the number of blocks that a miner creates in the latest $n_b$ blocks of blockchain. $n_{b}$ and $n_{othr}$ prevent a powerful miner from generating the majority of blocks of blockchain for controlling the blockchain, thus ensuring the decentralization. $n_{thr}$ is the threshold of the number of receipts that a miner should insert into its created block, which ensures sufficient receipts can be inserted into the blockchain. $TAG$ is a difficulty value, which is set to avoid the simultaneous creation of many valid blocks and thus helps reducing communication burdens. $\theta$ represents the time window that a miner waits for more valid blocks after receiving the first valid block.

**Algorithm 1** shows the creation of block $K$. $TX_{mr}$ means that the miner uses the age of $AM$ for making it become the creator of the next block. We denote the age of one token as the blockchain length from the block involving this token to the block $K$, and the miner uses tokens whose ages are the biggest. If $TX_e$ is inserted into blockchain, this suggests that a collector gives rewards to others and thus the age of related tokens goes to 0. $w(AM)$ represents the weight of age of $AM$ for adjusting the difficulty of the miner to create a valid block. The bigger the age of used $AM$ is, the easier the miner generates a valid block, since it is applied to control the difficulty of block generation. If a miner generates a valid block successfully,
and has not received valid blocks from others, it publishes its created block. When another miner receives this block, it can verify the validity of block by running Algorithm 1.

**Block Winner Selection**

Multiple valid blocks might be created at the same time. We design how to choose one of them as the block winner (i.e., the next block).

Aiming to summarize selection rules, Algorithm 2 shows determining the winner from two valid block candidates $B_{ca0}$ and $B_{ca1}$. They have two timestamps $t_{ca0}$ and $t_{ca1}$, and $L_{ca0}$ and $L_{ca1}$ receipts, respectively. In addition, both blocks’ creators have created $n_{pkca0}$ and $n_{pkca1}$ blocks within the latest $n_b$ blocks of blockchain. If $t_{ca0} 
eq t_{ca1}$, the block whose timestamp is earlier is considered as the winner for ensuring the efficiency of block creation. When $t_{ca0} = t_{ca1}$ and $n_{pkca0} 
eq n_{pkca1}$, the block whose creator has generated fewer blocks within the latest $n_b$ blocks of blockchain is regarded as the winner for preventing a miner from controlling the blockchain by generating the majority of blocks of blockchain. If $t_{ca0} = t_{ca1}$, $n_{pkca0} = n_{pkca1}$, and $L_{ca0} 
eq L_{ca1}$, the block including more receipts is the winner. When $t_{ca0} = t_{ca1}$, $n_{pkca0} = n_{pkca1}$ and $L_{ca0} = L_{ca1}$, the block whose related hash value $H_{pk}$ is smaller is regarded as the winner.

The valid blocks received by each miner might be different due to network latency. Thus, we show how to mitigate the inconsistency of block reception in order to avoid forking. If a miner receives the first valid block, it stops mining and sets the time window $\theta$ for receiving valid blocks from other miners. When a miner succeeds in generating a valid block but has not received valid blocks from other miners, it publishes its created block, waits $\theta$ for other valid blocks, and considers its block as the first received valid block. Because creating a valid block successfully before receiving other valid blocks can be considered as receiving the block ahead of the other blocks, $\theta$ can be adjusted and published by $N_{TI}$ due to network variations. The miner refuses to receive future blocks once $\theta$ expires. Finally, the miner performs block winner selection on all received valid blocks for determining the winner.

**Incentive Mechanism**

B4SDC should not only provide incentives for security data collection in route discovery and security data item collection, but also encourage the certificate and task issuer to periodically publish a task and miners to insert receipts and deposit messages into the blockchain.

In order to resist the collusion attacks of collector and intermediate nodes (i.e., intermediate nodes do not report their receipts deliberately), $N_r$’s public key and $TX_i’$’s $Q_{1i}$ recorded in blockchain can be used to construct a virtual tree [3]. Except the tree root, each non-leaf and leaf node obtains $\alpha$ and $\beta$ ($\alpha > \beta$) tokens from $pk_0$, respectively. As for each isolated node outside of the tree, the public address $pa$ gets $\alpha - \beta$ tokens, where $pa$ can be recorded in the genesis block. $P_{1i}$, $DT_{1i}$, and $SDI_{1i}$ can help constructing continuous virtual route paths originating from the first forwarding node in real routes. In the path, each node obtains $\alpha$ tokens from $pk_0$, but the last one only obtains $\beta$ tokens. Except all the nodes in the path, if there are $l$ remaining nodes in the real route that have reported involved receipts, $pa$ obtains $l \times (\alpha - \beta)$ tokens from $pk_0$ in order to avoid the collusion of intermediate nodes and collector [3]. In Fig. 3, if all the participants share their receipts and these receipts are inserted into the blockchain, a miner can construct a virtual tree with blue solid arrows based on RREQ forwarding, and two continuous virtual route paths based on RREP forwarding, namely $N_{21} \rightarrow N_{10} \rightarrow N_0$ and $N_d \rightarrow N_{11} \rightarrow N_0$. Therefore, $N_{10}$ and $N_1$ obtain $2 \alpha$ tokens, respectively. $N_{21}$ and $N_d$ obtain $\alpha$ tokens separately. $N_0$ can consider the two received RREPs as receipts, and thus obtains $2 \beta$ tokens. $N_{21}$ and $N_d$ obtain $\beta$ tokens respectively. Thus, $N_0$ should pay $6 \alpha + 4 \beta$ tokens for route discovery. In Fig. 4, when the receipts of all the participants are inserted into the blockchain, $N_{10}$ and $N_1$ obtain $2 \alpha$ and $3 \alpha$ tokens respectively, since $N_1$ separately forwards security data items from $N_{d1}$ and $N_{d2}, N_{d6}, N_{d1}$ and $N_{d2}$ obtain $\alpha$ tokens respectively. $N_v$ can use received $\{SDI_1, \phi_{11}\}$ and $\{SDI_{11}, \phi_{11}\}$ as receipts, thus obtaining $3 \beta$ tokens. As a consequence, $N_0$ should pay $8 \alpha + 3 \beta$ tokens for security data item collection.

When the first receipt with $UID$ or $DI$ is inserted in one block of blockchain, miners will not collect and insert receipts with the same $UID$ or $DI$ after lasting $n_r$ blocks from the block, thus redeeming the unexpended tokens of collector. $pa$ might have some tokens, which can be distributed to nodes whose receipts are inserted into blockchain in a specific distribution.

If the new $TX_v$, $TX_{de}$ or $TX_{cc}$ is inserted into the next block successfully, the sender of receipt or deposit message gives the relative miner a fixed ratio ($\omega, 0 < \omega < 1$) of its obtained rewards. The miner gives $\xi$ tokens to $N_{TI}$.

**4.4 Budget Balance of Collector**

In this part, we show the collector’s budget for route discovery and security data item collection.

**Budget for Route Discovery**

RREQ forwarding helps constructing a virtual tree. Since a budget is reserved in the whole route discovery, a collector can set the depth of the tree and the out-degree of each node at each hop for discovering a collection node.

$N_r$’s public key and $Q_{1i}$ received in route discovery can help constructing a virtual tree, and thus the collector can control the maximum depth $TTL$ of tree and the out-degree

---

**Algorithm 2: Block Winner Selection**

Input: $B_{ca0}$, $B_{ca1}$
Output: block winner
1. if $t_{ca0} \neq t_{ca1}$ then
2. Set the block whose timestamp is earlier as the winner;
3. else if $t_{ca0} = t_{ca1}$ then
4. if $n_{pkca0} \neq n_{pkca1}$ then
5. Set the block whose creator has created the fewer blocks within the latest $n_b$ blocks of blockchain as the winner;
6. else if $n_{pkca0} = n_{pkca1}$ then
7. if $L_{ca0} \neq L_{ca1}$ then
8. Set the block that includes more receipts as the winner;
9. else if $L_{ca0} = L_{ca1}$ then
10. Set the block whose related hash value is smaller as the winner;
11. end
12. end
13. end
14. end
15. return block winner;
$k_h$ of each node at each hop for balancing its budget in the whole route discovery. Conversely, it can determine the depth and out-degree due to a preset budget. Intuitively, the larger the out-degree and depth are, the higher the probability of discovering a collection node is. At the same time, this suggests a bigger budget. A tree with the largest RREQ forwarding scale is generated in route discovery when the receivers of all the nodes at each hop are different. The number of nodes at hop $h+1$ in the tree can be computed as $m_{h+1} = m_h k_h$. Given $TTL$, $k_h$, $h = 0, \ldots, TTTL - 1$, and $m_0 = 1$, the collector can obtain $m_{TTTL} = \prod_{h=0}^{TTTL-1} k_h$. When all the nodes at hop $TTTL$ have routes to a collection node, the payment of collector is the largest. We can compute the maximum payment in the whole route discovery as $CS_0 = m_{TTTL} \alpha + \sum_{h=0}^{TTTL-1} m_h \alpha + m_{TTTL}(TTTL - 1) \alpha + m_{TTTL} \beta = \prod_{h=0}^{TTTL-1} k_h (TTTL \beta + \sum_{h=1}^{TTTL-1} \prod_{i=0}^{h-1} k_i \alpha)$. Conversely, the collector can adjust the depth $TTL$ and out-degree $k_h$ for controlling the forwarding scale due to a given budget.

**Budget for Security Data Item Collection**

A collector can compute its maximum payment based on all discovered routes.

$N_v$’s public key, $DTI_v$ and $SDI_v$ can be adopted to construct continuous virtual route paths. When the collector succeeds in constructing security data from $N_{h_0}, \ldots, N_{h_d}$ by using the route paths $PA_{d_1}, \ldots, PA_{d_j}$, the virtual route paths are $PA_{d_1}, \ldots, PA_{d_j}$. If these route paths have only one point of intersection (namely the public key of collector), the collector’s payment for security data item collection is the largest. The collector can compute the maximum payment as $CS_1 = \sum_{i=0}^{n} (2h_{d_i} - 1) \alpha + \beta$, where $h_{d_i}$ is the hop count of $PA_{d_i}$. This budget can help the collector to deposit appropriate tokens for collection.

### 5 Analysis and Performance Evaluation

In this section, we analyze B4SDC, evaluate its performance, and compare it with mainstream blockchain and incentive systems.

#### 5.1 Analysis

In this part, we discuss that B4SDC provides incentives for all participants, and resists spoofing attacks, collusion attacks, and excessive forwarding and guarantees the fairness to all forwards in receipt collection. In addition, we analyze B4SDC in terms of the liveness, safety, fault tolerance, and decentralization of blockchain.

1) **Incentive**: B4SDC not only incents security data collection, but also encourages the certificate and task issuer to publish tasks and miners to insert more receipts or deposit messages into blockchain. In route discovery, each forwarder considers the received control information as receipts, adds sensed security data into the information, and then forwards this information. Each collection node generates and forwards a RREP after receiving a RREQ as a receipt. In security data item collection, each forwarder can also use the received information as receipts. After receiving a request as a receipt, each collection node generates and forwards a security data item. The forwarder and collection node can share received receipts to blockchain, and they obtain rewards if these receipts are inserted into blockchain. Miners are encouraged to insert more receipts and deposit messages into their created blocks, since they obtain more rewards from the reporters of receipt and deposit message when these receipts and deposit messages are added into blockchain successfully. The certificate and task issuer also obtains rewards from the miner of the next block since a block can be created only when a task is issued.

We analyze the acceptance of B4SDC by employing game theoretical analysis. The certificate and task issuer, collector, forwarders, collection nodes and miners are encouraged to cooperatively finish their works. The acceptance of B4SDC is analyzed based on a static game theory model. The players of game are the issuer, collector, forwarders and miners. We do not consider the situation of collection node, since it is the same as the forwarder’s. Suppose that these players know the information of each other and treat each other with the same priority. They adopt their own strategies in order to maximize their profits. We denote the strategy $S_x$ of issuer, collector, forwarder and miner as $ST_I, SCO, SFO$ and $SM$, respectively. $S_x = 1$ suggests that the entity $x$ cooperates with others and finishes its work honestly, and $S_x = 0$ means that $x$ refuses to cooperate.

First, we take into consideration $S_{CO} = 0$. The utility of collector is $ut_{CO} = 0$. If the issuer cooperatively publishes a task, its utility is $ut_{TI} = -c_{TI}$ since blocks cannot be created. $c_{TI}$ is the cost of issuer to generate a task. The utility of forwarder is $ut_{FO} = 0$, since no messages should be forwarded. The utility of miner is $ut_{M} = 0$, because no receipts and deposit messages are generated for block creation. If the issuer refuses to cooperate, the utility of issuer, forwarder and miner is $ut_{TI} = ut_{FO} = ut_{M} = 0$.

Second, we consider $S_{CO} = 1$. If $ST_I = 0$ and $SFO = 0$, the utility of collector, issuer, forwarder and miner is $ut_{CO} = -c_{CO}$, $ut_{TI} = 0$, $ut_{FO} = 0$ and $ut_{M} = 0$, respectively, because blocks cannot be created. $c_{CO}$ is the cost of collector to generate a RREQ and a valid deposit message. When $ST_I = 0$, $SFO = 1$ and $SM = 0$ or 1, the utility of collector, issuer, forwarder and miner is $ut_{CO} = b_{CO} - c_{CO}$, $ut_{TI} = 0$, $ut_{FO} = -c_{FO}$, and $ut_{M} = 0$ or $-n_{W}c_{M}$, respectively, $b_{CO}$ is the collector’s utility from route discovery or security data item collection, $c_{FO}$ is the cost of forwarder to forward a message and share a receipt. $n_{W}$ is the number of receipts collected by the collector, and $c_{M}$ represents the cost of collector to process a receipt. If $ST_I = 1$ and $SFO = 0$, the utility of collector, issuer, forwarder and miner is $ut_{CO} = -c_{CO}$, $ut_{TI} = -c_{TI}$, $ut_{FO} = 0$ and $ut_{M} = 0$. Because the forwarder refuses to forward messages and share receipts, and blocks cannot be generated successfully. When $ST_I = 1$, $SFO = 1$ and $SM = 0$, we can obtain the utility of collector, issuer, forwarder and miner is $ut_{CO} = b_{CO} - c_{CO}$, $ut_{TI} = -c_{TI}$, $ut_{FO} = -c_{FO}$ and $ut_{M} = 0$, since the miner does not create a block and thus the issuer and forwarder obtain no rewards. If $ST_I = 1$, $SFO = 1$ and $SM = 1$, the utility of collector, issuer, forwarder and miner is $ut_{CO} = b_{CO} - c_{CO} - CS * P_{CO}$, $ut_{TI} = \xi * P_{LI} - c_{TI}$, $ut_{FO} = \beta * (1-\omega) * P_{FO} - c_{FO}$, $ut_{M} = (\omega * CS - \xi) * P_{M} - n_{W}c_{M}$, $CS$ is the budget for route discovery or security data item collection, $P_{CO}$ represents the probability that all the receipts of involved nodes en route are inserted into the next block, $\xi$ is the
the out-degree.

Based on the above analysis, we can obtain an optimal solution. Nash Equilibrium is achieved if those four entities can conduct their work cooperatively (i.e., \( S_{co} = S_{T} = S_{fo} = S_{M} = 1 \)) and their utility exceeds 0. This conclusion can help setting \( CS, \omega \) and \( \xi \) and distributing rewards among issuer, forwarders, collection nodes and the block creator.

2) Resisting Spoofing Attacks: Each node cannot forge the message of other nodes for profits. To be specific, if \( N_{i} \) broadcasts a signed message and a neighbor receives this message, the neighbor attempts to forge another message signed by \( N_{i} \) for profits. However, it cannot forge the message successfully since we adopt the secure ECDSA for signatures and it cannot obtain the private key of \( N_{i} \) by solving a discrete logarithm problem.

3) Resisting Collusion Attacks: We discuss a route path constructed with RREQ forwarding. Due to the literature [3], if the last node colludes with the corresponding collector and does not report its receipt to blockchain, it obtains the behind-the-scene rewards \( \beta + \epsilon (\epsilon > 0) \) tokens from the collector. Thus, the collector is uncharged by \( \alpha - (\beta + \epsilon) \) tokens. In order to resist the collusion attack, the collector is required to transfer \( \alpha - \beta \) tokens to the public address \( pa \) when one receipt reporter is isolated from the route path. This suggests the collector pays extra \( \epsilon \) tokens, which refrains the collector from launching the collusion attack. Similarly, if one receipt reporter is isolated from the virtual tree that is constructed with RREQ forwarding and whose root is the collector’s public key, the collector pays \( \alpha - \beta \) tokens to \( pa \), which eliminates the collusion attack.

We consider a route path generated with RREP, DT, or SDI forwarding. For simplicity, we take the route path constructed with RREP forwarding as an instance. Except receipt reporters in the continuous route path originating from the collection node or an intermediate node having routes to this collection node, if there are \( l \) remaining receipt reporters in the real route (i.e., the route list \( RL \) in RREP), the collector is required to transfer \( l * (\alpha - \beta) \) tokens to the public address. Similarly, this can avoid the collusion attack since the tokens that the collector transfers to \( pa \) for the attack are larger than its attack cost.

When the collector \( N_{0} \) receives a valid RREP or security data item en route, \( N_{0} \) is not allowed to consider its received messages as receipts. Because it is not willing to share its receipt to the blockchain for saving the cost of reporting a receipt and reducing the payment to \( N_{1} \) by \( \alpha - \beta \) tokens. \( N_{1} \) can obtain at most \( \beta \) tokens. In order to guarantee the benefit of \( N_{1} \), a common neighbor of \( N_{0} \) and \( N_{1} \) collects the RREP or security data item as a receipt, and shares its receipt to the blockchain. Therefore, \( N_{1} \) might obtain at most \( \alpha \) tokens from \( N_{0} \) and its profits can be guaranteed.

4) Resisting Excessive Forwarding: Due to a budget (i.e., the maximum payment) in the whole route discovery, a collector can set the maximum depth \( TTL \) of virtual tree and the out-degree \( k_{h} \) of each node at each hop for discovering a collection node. In route discovery, a collector constrains the scale of RREQ forwarding by adding \( TTL \) and \( k_{h} \) into this RREQ. After receiving the RREQ, each forwarder should select a fixed number of RREQ receivers, otherwise, it obtains no rewards since its neighbor shares a report to blockchain for rewards when discovering its misbehaviors. Even if a node that is not specified as a receiver or whose hop count exceeds \( TTL \) can obtain the RREQ, it cannot use the received RREQ as a receipt for getting rewards from the collector since each receipt should be verified and accepted by a majority of all miners. Therefore, B4SDC resists excessive forwarding.

5) Fairness: As long as a node (except collector) participates in receipt generation, it can save its received messages as receipts, and is willing to share these receipts to blockchain for rewards. The issuer is encouraged to publish a task for rewards. Miners are incented to insert more receipts into their created blocks for rewards. Thus, more receipts are inserted into the next block. As a result, as long as a node (except collector) participates in receipt generation, it can obtain rewards with a high probability, which ensures the fairness for each forwarder and collection node.

6) Liveness: The liveness of B4SDC can be ensured, namely the probability of successfully creating a valid block for the next block tends to 1 when a task is published. If the blockchain is comprised of \( N \) blocks and the certificate and task issuer publishes the \( (N + 1) \)-th task, we compute the probability that at least one valid is generated for the next block until the reference time \( T_{Sr} \).

After the issuer publishes the task, the miner \( M_{i}, i = 1, \ldots, CN \) starts to create valid blocks at the time \( T_{S} \) until \( T_{Sr} \). We suppose \( T_{S1} \leq \cdots \leq T_{S_{CN}} \leq T_{Sr} \) due to the existing network latency. The block creation is a random event, thus the time that \( M_{i} \) spends creating a valid block successfully follows an exponential distribution [20]. The probability of \( M_{i} \) to successfully create a valid block from \( T_{S_{i}} \) to \( T_{Sr} \) can be computed as \( 1 - e^{- (T_{Sr} - T_{S_{i}})/(\mu_{i})} \), where \( \mu_{i} \) is the average time that \( M_{i} \) spends creating a valid block successfully. Note that \( w(AM) \) is used to adjust the difficulty value \( TAG \), thus \( \mu_{i} \) of some miners is short. The probability of these \( CN \) miners to succeed in creating at least one valid block is

\[
P_{Li} = 1 - \prod_{i=1}^{CN} \left( 1 - [1 - e^{- (T_{Sr} - T_{S_{i}})/(\mu_{i})}] \right)
\]

If \( \sum_{i=1}^{CN} (T_{Sr} - T_{S_{i}})/(\mu_{i}) \) is increased, \( P_{Li} \) tends to 1. In order to increase \( \sum_{i=1}^{CN} (T_{Sr} - T_{S_{i}})/(\mu_{i}) \), we can grow the number \( CN \) of participating miners or \( T_{Sr} - T_{S_{i}} \) by growing \( T_{Sr} \). As long as a miner with short \( \mu_{i} \) participates, \( P_{Li} \) tends to 1 rapidly with \( T_{Sr} \) increasing, thus at least one valid block can be generated within short time.

7) Safety: B4SDC ensures the safety of blockchain, namely the block winner consistency can be achieved among honest miners for avoiding the forks of blockchain. Due to the literature [19], we can compute the probability of a miner in the network to receive the valid block with the earliest timestamp as at least \( 1 - e^{-\theta /\lambda} \), where \( \lambda \) is the average network latency. Therefore, at least \( 1 - e^{-\theta /\lambda} \) of all the
miners in the network receive the block during their preset time window $\theta$. When $\theta$ is large enough, $1 - e^{-\theta/\lambda}$ approximates to 1, which suggests almost all the miners can receive the block with the earliest timestamp. Therefore, the block winner consistency among honest miners can be ensured. The detail process refers to the literature [19].

8) Fault Tolerance: Although an attacker controls $\eta \leq 50\%$ miners, it cannot destroy the consensus that honest miners have reached and ongoing consensus. Except the miners controlled by the attacker (i.e., malicious miners), other miners are assumed to behave independently and maximize their profits rationally, which are called honest miners.

If the block $B_K$ has got the consensus of honest miners, another block $B'_{K'}$ is stored by the malicious miners. The malicious miners generate a new block based on $B_K$. It is possible that $B'_{K'}$ is superior to $B_K$, since the malicious miner might not publish $B_K$. Suppose $B_K$ and $B'_{K'}$ are generated based on the block $B_{K-1}$ of blockchain. If a new miner participates in the blockchain system, it requests blocks from miners in order to maintain blockchain. The miner could receive $B_K$ and $B'_{K'}$ since the honest and malicious miners exist in the system. Therefore, it should determine which block should be stored as the valid block of blockchain. In order to solve this problem, the miner can request blocks from a number of miners, and get the number of received $B_K$ and $B'_{K'}$ that is denoted as $n_{B_K}$ and $n_{B'_{K'}}$, respectively. If $n_{B_K} \geq n_{B'_{K'}}$, the miner stores $B_K$. Otherwise, it stores $B'_{K'}$. When $n_{B_K} + n_{B'_{K'}}$ is large enough, $n_{B'_{K'}}/(n_{B_K} + n_{B'_{K'}})$ approximates to $\eta$, thus $n_{B'_{K'}}/(n_{B_K} + n_{B'_{K'}}) \leq 50\%$. As long as the miner requests blocks from a large enough number of miners, it stores $B_K$ with the probability $1 - \eta \geq 50\%$. Therefore, the number of honest miners storing $B_K$ is always larger than the number of malicious miners storing $B'_{K'}$. Thus the attacker cannot destroy the consensus that the honest miners have reached.

When a malicious miner generates a valid block earlier than honest miners, it could not immediately publish the block, but waits some time and then publishes the block in order to cause the forks of blockchain and perturb the ongoing consensus. However, the malicious miner cannot succeed in perturbing the consensus. It is noted that the time of miners is synchronized, each block includes the timestamp of its creation, and all the miners are monitored by their neighbors. If the malicious miner publishes the malicious block, its neighbors could refuse to forward the block due to its invalid timestamp. Even if the honest miners could receive the block within their time period $\theta$, they can use lightweight external methods to detect the invalid timestamp and reject the block. For instance, wormhole attack detection might help estimating the real time of block creation. In order to guarantee the accuracy of estimation, multiple routes that include the different neighbors of malicious miner can be adopted for block propagation, thus solving the low trust problem of estimation of one route. Therefore, the attacker cannot perturb the ongoing consensus when it controls $\eta \leq 50\%$ of all the miners.

In summary, B4SDC resists the misbehaviors of 50% of all the miners.

9) Decentralization: B4SDC prevents a powerful miner from creating the majority of blocks of blockchain for controlling the blockchain, thus ensuring the decentralization. That is, even if the miner creates a valid block more efficiently than others, we can reduce its probability of succeeding in creating the next block to a low degree.

Due to the literature [19], we can model B4SDC including a powerful miner as a state machine, and then define system states. Additionally, we can derive the probability of system state transition and the probability of the system to reach each state. When $n_{\text{other}} = 1$, the probability of the powerful miner to generate the next block can be computed as $\rho/(1 + \rho n_b)$, where $\rho$ represents the probability that the powerful miner creates a valid block earlier than others. When the powerful miner always generates a valid block earlier than others, $\rho$ approximates to 1, and the probability of this miner to succeed in generating the next block is $1/(1 + n_b)$, which decreases with $n_b$ increasing. As a consequence, we can fix $n_{\text{other}}$, and then select the appropriate $n_b$ for reducing the probability of the powerful miner to successfully create the next block to a low degree, thus ensuring the decentralization of blockchain. The detail process refers to the literature [19].

5.2 Performance Evaluation
In this part, we evaluated B4SDC through simulations.

Metrics: (1) Message forwarding time: The average message forwarding time of node at each hop; (2) Message size: The average size of message that a node at each hop forwards; (3) Transaction confirmation time: The average time of B4SDC to confirm a receipt or deposit message; (4) Throughput: The average number of receipts and deposit messages that are successfully recorded in blockchain per second; (5) Block size; (6) Memory usage: The average memory usage of node to forward a message and create and verify a block.

Experimental Settings: With respect to experimental environments, we simulated B4SDC using NS3 and C++ in a laptop that runs Ubuntu 18.04 with Intel Core i5-6300HQ CPU @2.3Ghz and 12GB memory. We simulated a node as the certificate and task issuer, 1000 nodes for security data collection, and 4 miners. In terms of network settings, the propagation time of bitcoin block between two miners follows an exponential distribution [20]. Thus, we simulated a real network by introducing the network delay between two miners that follows the exponential distribution with the rate parameter $\lambda = 12.6$s. With respect to receipt generation, we adopted the DSR route discovery and used discovered routes to transmit the request of collector and the security data of collection nodes. In route discovery, a node at each hop adds its local timestamp (i.e., a kind of security data) into received RREQs and RREPs. In security data item collection, each collection node uses a row of feature values in the NSL-KDD database as security data.

Experimental Results: Message Forwarding Time: As shown in Fig. 5(a), we can observe the average message forwarding time of a node at each hop. The time does not change with the increase of hops. The forwarding time of RREQ, RREP, request, and security data item of the node is approximately 20ms, 15ms, 25ms and 35ms, respectively. In receipt sharing, we tested the time token by the node to share a receipt.
When the receipt is a RREQ, RREP, request or security data item, the time cost of the node to share the receipt is equal to the time cost of the node to forward it.

**Message Size:** We show the average size of a message that a node at each hop forwards in Fig. 5(b). When the number of hops increases, the size of RREQ increases but the size of RREP decreases, because the node at hop \( h \) adds its sensed security data into the RREQ received from the node at hop \( h - 1 \) and the RREP received from the node at hop \( h + 1 \). The size of request and security data item remains unchanged around 0.8KB and 1.4KB, respectively. When a node considers a received RREQ as a receipt, the size of its shared receipt equals that of received RREQ. If a node regards a received RREP as the receipt, the size of receipt is equal to that of received RREP. When the receipt is a received request and security data item, the size of each node’s shared receipt is 0.8KB and 1.4KB, respectively.

**Transaction Confirmation Time:** As shown in Fig. 5(c), we observe the transaction confirmation time when the task frequency is set to 4 tasks per minute, \( n_b = 0 \), and we change the number \( n = L + X + Y \) of receipts and deposit messages that should be inserted into the next block. The time increases linearly with \( n \) increasing. The bigger \( n \) suggests a miner verifies more receipts and deposit messages of a received block, which takes more time.

If \( n_b = 0 \) and \( n = 1000 \), we show the transaction confirmation time by changing the task frequency in Fig. 5(d). When the task frequency is less than 1.7 tasks per minute, the time decreases sharply with the task frequency increasing. Because the task interval is longer than the average consensus time and is regarded as the transaction confirmation time, and the task interval is inversely proportional to the task frequency. If the task frequency exceeds 1.7 tasks per minute, the transaction confirmation time remains invariant since the task interval is shorter than the consensus time.

When we set the task frequency to 4 tasks per minute, \( n_bhr = 1 \) and \( n = 1000 \), TABLE 2 shows the transaction confirmation time by changing \( n_b \). The time grows with the increase of \( n_b \), since fewer miners participate in mining.

**Throughput:** When the task frequency is set to 4 tasks per minute and \( n_b = 0 \), we observe the throughput by changing \( n \). As shown in Fig. 5(e), the throughput increases linearly when \( n \) increases. Because more receipts and deposit messages are inserted into the next block.

If \( n_b = 0 \) and \( n = 1000 \), we show the throughput by changing the task frequency. As depicted in Fig. 5(f), if the task frequency increases from 1.1 to 1.7 tasks per minute, the throughput grows linearly, since the task interval is longer than the consensus time and thus inversely proportional to

#### TABLE 2

<table>
<thead>
<tr>
<th>( n_b )</th>
<th>TC (s)</th>
<th>TH (tx/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>36.9</td>
<td>37.3</td>
</tr>
<tr>
<td>1</td>
<td>37.5</td>
<td>38.1</td>
</tr>
<tr>
<td>2</td>
<td>37.6</td>
<td>38.2</td>
</tr>
<tr>
<td>3</td>
<td>37.6</td>
<td>38.2</td>
</tr>
</tbody>
</table>

TC: transaction confirmation time; TH: throughput.

#### TABLE 3

<table>
<thead>
<tr>
<th>Operations</th>
<th>Memory usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwarding a RREQ or sharing a receipt (i.e., received RREQ)</td>
<td>18.63MB</td>
</tr>
<tr>
<td>Forwarding a RREP or sharing a receipt (i.e., received RREP)</td>
<td>18.75MB</td>
</tr>
<tr>
<td>Forwarding a request or sharing a receipt (i.e., received request)</td>
<td>18.59MB</td>
</tr>
<tr>
<td>Forwarding a security data item or sharing a receipt (i.e., received security data item)</td>
<td>18.71MB</td>
</tr>
<tr>
<td>Creating a block</td>
<td>421.98MB</td>
</tr>
<tr>
<td>Verifying a block</td>
<td>421.98MB</td>
</tr>
</tbody>
</table>
the task frequency. If the task frequency exceeds 1.7 tasks per minute, the throughput is almost invariable. Because the task interval is shorter than the consensus time.

When \( n_{\text{thre}} = 1 \), \( n = 1000 \) and the task frequency is 4 tasks per minute, we observe the throughput with different \( n_b \). As shown in TABLE 2, when \( n_b \) increases, the throughput decreases due to the longer transaction confirmation time.

**Block Size:** Fig. 5(g) shows the block size with different \( n \). The block size increases linearly with the increasing of \( n \), since the larger \( n \) suggests more receipts and deposit messages are added into blockchain.

**Memory Usage:** We observe the memory usage of node when some operations are performed, as shown in TABLE 3. In route discovery and security data item collection, the memory usage of the node to forward a RREQ and RREP is 18.63MB and 18.75MB, respectively. In receipt sharing, the memory usage of the node to share a receipt (i.e., RREQ, RREP, request and security data item) is also 18.63MB, 18.75MB, 18.59MB and 18.71MB, respectively. The memory size of current routers is 64MB or 128MB, even as large as 518MB. Thus, many smart devices have a sufficient memory space to forward a RREQ, RREP, request and security data item and share a receipt. On the other hand, the memory size of a miner should be at least 421.98MB. Since the memory size of current smart phones reaches 8GB, running B4SDC is not heavy in mobile phone organized MANETs.

**Comparison:** In TABLE 4, we compare B4SDC with mainstream blockchain and incentive systems in terms of incentive, security and performance.

Some blockchain systems do not consider incentives, e.g., Algorand using BFT. The liveness and safety of PoW, PoUW, BFT, PoS, and trees and DAGs based blockchain systems are analyzed [12], [36]. PoW, PoUW and PoS suffer from the outsourceability of task [37], thus not ensuring the decentralization. However, B4SDC guarantees the decentralization, since it can prevent a powerful miner from creating the majority of blocks of blockchain by setting appropriate \( n_b \) and \( n_{\text{thre}} \). In the simulation, if the task frequency is 4 tasks/min and \( n_b = 0 \), the transaction confirmation time and throughput are \( TC = 36.9s \) and \( TH = 27.1tx/s \). They are rational since a miner consumes almost no computing resources for solving a puzzle. Suppose there exist \( M \) miners. In PoW, BFT, PoS, PoUW and trees and DAGs based blockchain systems and B4SDC, a message should be shared to all miners for making them insert this message into blockchain, thus the communication complexity is \( O(M) \).

In PoW, PoUW, PoS, and trees and DAGs based blockchain systems and B4SDC, the next block should be sent to all the miners. Therefore, their communication complexity is \( O(M) \). BFT requires information exchange between any two miners, thus its communication complexity is \( O(M^2) \).

The scalability of blockchain is its ability to maintain a high throughput when more miners participate [38]. We assume that there are \( M \) miners in blockchain systems. In B4SDC, each miner can collect receipts and deposit messages before the certificate and task issuer publishes a task. Some miner receives the task, and quickly creates the next block with a negligible difficulty based on collected \( n \) receipts and deposit messages. The time of task propagation between the issuer and each miner is negligible since the size of task is small. The miner publishes the next block. For simplicity, we consider the network composed of the \( M \) miners. The average network delay of a receipt or deposit message between any two miners increases logarithmically with \( M \) increasing [39]. Thus, the average time of block propagation between any two miners can be estimated as \( n\log_2 M \), where \( \tau \) is a parameter related to network bandwidth. Signature verifications are the most time-consuming operations in verifying the validity of a block, and the verification time of each signature is denoted as \( T_v \). When the task interval is shorter than the consensus time, we estimate the throughput of B4SDC as \( \frac{TH_0}{n(\log_2 M + nT_v)} = \frac{1}{n\log_2 M + T_v} \). When \( M \) increases, \( TH_0 \) decreases. When the task interval is longer than the consensus time, we can obtain the throughput of B4SDC as \( TH_1 = \frac{n}{TN} \), where \( TN \) represents the task interval. BFT requires 3-round information exchanges. When there are \( M \) miners in a BFT based blockchain system, the average time of 3-round information exchanges is estimated as \( (n+2)\log_2 M^2 \).

**Comparison:** In TABLE 4, we compare B4SDC with mainstream blockchain and incentive systems in terms of incentive, security and performance.

Some blockchain systems do not consider incentives, e.g., Algorand using BFT. The liveness and safety of PoW, PoUW, BFT, PoS, and trees and DAGs based blockchain systems are analyzed [12], [36]. PoW, PoUW and PoS suffer from the outsourceability of task [37], thus not ensuring the decentralization. However, B4SDC guarantees the decentralization, since it can prevent a powerful miner from creating the majority of blocks of blockchain by setting appropriate \( n_b \) and \( n_{\text{thre}} \). In the simulation, if the task frequency is 4 tasks/min and \( n_b = 0 \), the transaction confirmation time and throughput are \( TC = 36.9s \) and \( TH = 27.1tx/s \). They are rational since a miner consumes almost no computing resources for solving a puzzle. Suppose there exist \( M \) miners. In PoW, BFT, PoS, PoUW and trees and DAGs based blockchain systems and B4SDC, a message should be shared to all miners for making them insert this message into blockchain, thus the communication complexity is \( O(M) \). In PoW, PoUW, PoS, and trees and DAGs based blockchain systems and B4SDC, the next block should be sent to all the miners. Therefore, their communication complexity is \( O(M) \). BFT requires information exchange between any two miners, thus its communication complexity is \( O(M^2) \).

The scalability of blockchain is its ability to maintain a high throughput when more miners participate [38]. We assume that there are \( M \) miners in blockchain systems. In B4SDC, each miner can collect receipts and deposit messages before the certificate and task issuer publishes a task. Some miner receives the task, and quickly creates the next block with a negligible difficulty based on collected \( n \) receipts and deposit messages. The time of task propagation between the issuer and each miner is negligible since the size of task is small. The miner publishes the next block. For simplicity, we consider the network composed of the \( M \) miners. The average network delay of a receipt or deposit message between any two miners increases logarithmically with \( M \) increasing [39]. Thus, the average time of block propagation between any two miners can be estimated as \( n\log_2 M \), where \( \tau \) is a parameter related to network bandwidth. Signature verifications are the most time-consuming operations in verifying the validity of a block, and the verification time of each signature is denoted as \( T_v \). When the task interval is shorter than the consensus time, we estimate the throughput of B4SDC as \( \frac{TH_0}{n(\log_2 M + nT_v)} = \frac{1}{n\log_2 M + T_v} \). When \( M \) increases, \( TH_0 \) decreases. When the task interval is longer than the consensus time, we can obtain the throughput of B4SDC as \( TH_1 = \frac{n}{TN} \), where \( TN \) represents the task interval. BFT requires 3-round information exchanges. When there are \( M \) miners in a BFT based blockchain system, the average time of 3-round information exchanges is estimated as \( (n+2)\log_2 M^2 \).

The throughput of BFT based blockchain system can be estimated as \( TH_2 = n/[(n+2)\log_2 M^2 + nT_v] \). Therefore, if the task interval is shorter than the consensus time in B4SDC, the throughput of the BFT based blockchain system is lower compared to the B4SDC’s throughput. When the task interval is longer than the consensus time in B4SDC and \( M > \frac{(TN-nT_v)}{2(n+2)} \), the throughput of the BFT based blockchain system is lower than that of B4SDC. When \( M (M > \frac{(TN-nT_v)}{2(n+2)}) \), the throughput of B4SDC decreases more slowly than that of the BFT based blockchain system, which implies the higher ability of B4SDC to ensure a high throughput. Therefore, B4SDC can support better scalability compared to BFT based blockchain systems. In the blockchain systems based on PoW, PoUW, and trees and DAGs, the time of block creation \( T_c \) is very long since they aim to prevent an attacker from easily creating blocks in order to avoid forking. Their throughput can be denoted as \( TH_3 = n/(T_c + n\log_2 M + nT_v) \). Based on the above analysis, we can see that their scalability is worse than the scalability of B4SDC with large enough \( M \) since \( T_c \) is very large. In a PoS based blockchain system, some miner can quickly create a block with a negligible difficulty. The throughput of the PoS based blockchain systems is \( TH_4 = 1/(\log_2 M + T_v) \). Thus, \( TH_0 \) and \( TH_4 \) are equal. When the task interval is longer than the consensus time in B4SDC and \( M > \frac{(TN-nT_v)}{n} \), the throughput of B4SDC is higher than that of the PoS based blockchain system. If \( M (M > \frac{(TN-nT_v)}{n}) \), the ability of B4SDC to maintain a high throughput is higher than the PoS based blockchain systems. Therefore, the scalability of B4SDC is better than that of the PoS based blockchain system when \( M \) is large enough. As a consequence, B4SDC supports better scalability compared to PoW, BFT, PoUW, PoS, and trees and DAGs based blockchain systems.

In summary, B4SDC provides incentives and solves the three challenges (i.e., forking, low efficiency and the trend of centralization) of mainstream blockchain systems, and the security problems of mainstream incentive systems. In addition, B4SDC ensures better scalability compared to the mainstream blockchain systems. Therefore, B4SDC outperforms the mainstream works.
In this paper, we proposed B4SDC, a blockchain system for security data collection in MANETs. It not only incented security data collection in route discovery, but also motivated the collection at collection nodes. Analysis and simulation based experiments showed that B4SDC can resist spoofing attacks, collusion attacks, and excessive forwarding, guarantee fairness as much as possible, and also solve the main problems of current blockchain technologies, namely forking, low efficiency, and centralization. In the future, we will explore a feasible scheme to preserve privacy in B4SDC.

### 6 Conclusion

In this paper, we proposed B4SDC, a blockchain system for security data collection in MANETs. It not only incented security data collection in route discovery, but also motivated the collection at collection nodes. Analysis and simulation based experiments showed that B4SDC can resist spoofing attacks, collusion attacks, and excessive forwarding, guarantee fairness as much as possible, and also solve the main problems of current blockchain technologies, namely forking, low efficiency, and centralization. In the future, we will explore a feasible scheme to preserve privacy in B4SDC.

### Acknowledgments

The work is supported in part by the National Natural Science Foundation of China under Grants 61672410, 61802293 and 11576202, the Academy of Finland under Grants 308087 and 314203, the Key Lab of Information Network Security, Ministry of Public Security under grant No. C18614, the National Postdoctoral Program for Innovative Talents under grant BX20180238, the Project funded by China Postdoctoral Science Foundation under grant CLDL-20182119, the Shaanxi Institute of China Electronics Technology Group Corporation, P.R. China under grant CLDL-20182119, the Shaanxi Institute of China Electronics Technology Group Corporation, P.R. China under grant CLDL-20182119, and the Shaanxi Institute of China Electronics Technology Group Corporation, P.R. China under grant CLDL-20182119, and the Shaanxi Institute of China Electronics Technology Group Corporation, P.R. China under grant CLDL-20182119.

### References


Zheng Yan received the BEng degree in electrical engineering and the MEng degree in computer science and engineering from the Xi’an Jiaotong University, Xi’an, China in 1994 and 1997, respectively, the second MEng degree in Information security from the National University of Singapore, Singapore in 2000, and the licentiate of science and the doctor of science in technology in electrical engineering from Helsinki University of Technology, Helsinki, Finland. She is currently a professor at the Xidian University, Xi’an, China and a visiting professor at the Aalto University, Espoo, Finland. Her research interests are in trust, security, privacy, and security-related data analytics. Prof. Yan serves as a general or program chair for 30+ international conferences and workshops. She is also an associate editor of many reputable journals, e.g., IEEE Internet of Things Journal, Information Sciences, Information Fusion, JNCA, IEEE Access, SCN, etc.

Xiaokang Zhou (M’12) received the Ph.D. degree in human sciences from Waseda University, Japan, in 2014. From 2012 to 2015, he was a research associate with the Department of Human Informatics and Cognitive Sciences, Faculty of Human Sciences, Waseda University, Japan. From 2016, he has been a lecturer with the Faculty of Data Science, Shiga University, Japan. He also works as a visiting researcher in the RIKEN Center for Advanced Intelligence Project (AIP), RIKEN, Japan, from 2017. Dr. Zhou has been engaged in interdisciplinary research works in the fields of computer science and engineering, information systems, and social and human informatics. His recent research interests include ubiquitous computing, big data, machine learning, behavior and cognitive informatics, cyber-physical-social-system, cyber intelligence and cyber-enabled applications. Dr. Zhou is a member of the IEEE CS, and ACM, USA, IPSJ, and JSAI, Japan, and CCF, China.

Shohei Shimizu is a Professor at the Faculty of Data Science, Shiga University, Japan and leads the Causal Inference Team, RIKEN Center for Advanced Intelligence Project. He received a Ph.D. in Engineering (Statistical Science) from Osaka University in 2006. His research interests include statistical methodologies for learning data generating processes such as structural equation modeling and independent component analysis and their application to causal inference. He received Hayashi Chikio Award (Excellence Award) from the Behaviormetric Society in 2016. He is a coordinating editor of Springer Behaviormetrika since 2016.

Huidong Dong is currently pursuing the master’s degree with the School of Cyber Engineering, Xidian University. His research interests include IoT security and applied cryptography.

Gao Liu is currently a Ph.D. candidate in Information Security, School of Cyber Engineering, Xidian University. His research interest includes network security measurement, data collection, blockchain, deep learning, machine learning, e-voting, e-lottery, micropayment, data aggregation in Smart Grid, and authentication in Smart Grid and VANETs.