Blockchain-based secure multi-resource trading model for smart marketplace

Yakubu, Bello Musa; Khan, Majid I.; Javaid, Nadeem; Khan, Abid

Published in:
Computing

DOI:
10.1007/s00607-020-00886-7

Publication date:
2021

Citation for published version (APA):
Yakubu, B. M., Khan, M. I., Javaid, N., & Khan, A. (2021). Blockchain-based secure multi-resource trading model for smart marketplace. *Computing*, *103*(3), 379-400. https://doi.org/10.1007/s00607-020-00886-7

Computing (2021) 103:379–400
https://doi.org/10.1007/s00607-020-00886-7

SPECIAL ISSUE ARTICLE

Blockchain-based secure multi-resource trading model for smart marketplace

Bello Musa Yakubu[1] · Majid I. Khan[1] · Nadeem Javaid[1] · Abid Khan[2]

Received: 14 September 2020 / Accepted: 1 December 2020 / Published online: 11 January 2021

Abstract
Developments in sensors and communication technology lead to the emergence of smart communities where diverse collaborative applications can be enabled. One such application is the Smart Market Place (SMP), where participants of the smart community can trade resources, such as energy, internet bandwidth, water, etc., using a virtual currency (such as ether). However, most of the existing SMP trading models are proposed to trade a single resource and also restrict a participant to perform only a single transaction at a time. Restriction on multiple parallel transactions is imposed to protect the participants against the double-spending attack in the SMP. This work proposes a secure multi-resource trading (SMRT) model that is based on public Ethereum blockchain. SMRT allows participant of a SMP to trade multiple resources and initiate parallel transactions. Moreover, detailed security analysis and adversary model are presented to test the effectiveness and to assess the resilience of the proposed model against the double-spending attack. The adversary model is based on partial progress towards block production which is influenced by time advantage and average computing power. Furthermore, simulation based analysis and comparison of SMRT is also presented in terms of security, performance, cost and latency of transactions. It is observed that SMRT not only provides protection against the double spending attack, but it also reduces the computational overhead of the proposed model up to 50% as compared to existing trading models.

Keywords Blockchain technology · Smart marketplace · Multi-resource trading · Double-spending attack

Mathematics Subject Classification 68M25 · 68P27

1 Introduction

Advancements in sensor integrated devices and network innovations has led to the emergence of smart communities where resources, such as energy, water, and internet

Extended author information available on the last page of the article

bandwidth can be traded among the participating entities for example local energy smart grid (LESM) system and community water supply cooperation (CWSC), this type of trading platform is named as Smart Market Place (SMP) [1,2]. Depending on the architecture, existing SMP solutions can be categorized into two types, centralized and decentralized. The centralized approach is based on a traditional contemporary model, where the system provides a central control for processing, administering, and delivering resource trades [3]. The approach has some benefits for example, ease of set-up and flexible overall system control. However, it may face a range of privacy and security problems, such as central point of failure, controller compromise, and information leakage [4]. Alternatively, distributed approach enables resource management and exchange among the participants with either fully or semi-distributed mechanisms [5,6]. Using the distributed approach, a participant can obtain required resources from its neighborhood without or partially relying on a central resource trading manager [7]. In distributed approach, household owners with surplus resources achieve greater profits compared to the centralized scheme, because it enables participants to express and settle resource prices, thus establishing a competitive resource trading market [6,8]. Additionally, the distributed approaches help to overcome the limitations of centralized approaches, such as central point of failure and high influence of central authorities in terms of control and management [5].

Most of the existing SMP solutions are decentralized in nature (implemented using blockchain technology, such as Ethereum) and allow only one type of resource to be traded (for example, energy or water) [9,10]. Moreover, a participant can trade with only one other participant at a given time, that is, it is not permitted for a seller to trade with two or more buyers in parallel. For example, [11,12] introduce trading models to enable data/internet bandwidth trading, and [4,13] are solemnly presented to enable energy trading, only allowing one trading transaction at a time. We refer to these types of trading platform as one-to-one single resource type (OTOSRT) approaches. These type of approaches are established to ensure transparency and to avoid double spending behaviours by participants during the trading activities. As a result, existing SMP models are not applicable in scenarios where a seller wants to divide a large quantity of resources into small portions and publish it to multiple buyers to sell in parallel. Moreover, a seller may desire to trade different types of resources, such as energy, water, and internet bandwidth. We refer such a scenario as one-to-many multiple resource type (OTMMRT) trading model that needs to be developed. Moreover, the OTMMRT facility in SMP can be exploited by the adversary to launch a double spending attack, that needs to be addressed. Two types of double spending attacks for these scenarios are, token based double spending, and ownership based double spending [4,14]. In token based double spending, the attacker is an authenticated requester node who wants to spend a single digital token more than once in several trading transactions. While, in ownership based double spending, the attacker is an authenticated provider node who wants to sell ownership of a resource (e.g. Energy) to two or more buyers.

Considering the significance for achieving scalable and secured resource trading, the need for OTMMRT trading model is evident. In this paper, we present a novel OTMMRT trading model with low overhead and latency that also provides protection from double-spending attacks.

Our contributions Following contributions are made in this paper:

- a scalable OTMMRT trading scheme is proposed. It enables a participant to perform multi-resource trading with one or more participants simultaneously,
- a security feature to handle ownership based double spending attack together with a token based double spending attack in SMP had been proposed,
- the smart contract architecture was built, tested, and carefully evaluated to ensure that there are adequate flexibility and defense against severe security threats, such as re-entrancy vulnerability, timestamp dependency, transaction dependency and parity multisig bug,
- detailed security analysis was carried out to show that the SMRT scheme is protected against specified adversary model. Experiments had been performed to compare the effectiveness of the SMRT model compared to existing schemes.

Organization of the paper The rest of the paper is organized as follows: Sect. 2 provides an overview of the related work. In Sect. 3, we present the system model, Sect. 4 describes SMRT trading scheme. Section 5 discusses the security analysis, Sect. 6 explains the testing and evaluation, and Sect. 7 is the conclusion.

2 Related work

Most of the existing SMPs are designed for OTOSRT [11,12]. This is to curtails double spending attacks which may be inevitable [15]. However, there are very few schemes such as [4] that allow trading a single resource with two or more participants in parallel. Yet, these techniques face many challenges when it comes to deployment in a real world scenario, such as identity and privacy issues, and double spending attacks. Similarly, technique proposed in [16] is developed for OTOSRT, but it can be used to trade various resource types (e.g. energy, water etc.) during different trading periods.

The existing trading schemes can also be categorised based on their architectures that is centralized, semi-distributed, or fully distributed [15]. Initially, some of the schemes uses centralized contemporary architecture [1,3] when determining how and when to charge a customer based on token payment approach. The resulting imprecision, may not be acceptable in the event of higher and less regular payments. This may be as a result of the control center for processing, managing and distributing resources and token payment transactions. The approach has several benefits, e.g. easy set-up and efficient management; yet, it still poses wide range of privacy and security issues similar to several other centralized approaches [1,4].

The token based payment system can be used not only in conventional centralized client trading schemes, but also in semi-distributed P2P platforms [17,18]. The concept of private SMP P2P trading and payment was proposed in [16,18], the approach ensures that communications and transactions remain private. This original concept demanded an on-line manager to confirm tokens before vendors could provide their services to protect them from double spending. Technique [19,20] uses digital tokens as a payment mechanism for P2P transactions, taking advantage of the fact that peers are clients and traders at the same time. Buyers will then pay with the (exchangeable)

tokens they receive from selling their own goods, reducing the amount of contacts with the central authority. Similarly, the scheme [21,22] broadens the concept of [19] and guarantees that tokens are anonymous as well as the responsibility of assigning authority to peers themselves. In addition, technique [21] proposes a method for real-time double spending detection that uses the P2P network as a semi-decentralized database for spent tokens and queries using a DHT routing layer such as [23]. Besides, techniques [1,24–26] addresses the same concept in more detail and tests various situations for the position of collected, spent tokens. Yet, neither solution could provide tough assurances against double spending, particularly when a portion of the P2P nodes are compromised [27]. The semi-decentralized database cannot be completely trusted except if the secure routing and integrity of peers is assured and can only accommodate probabilistic assurances.

Notwithstanding, better solution for token-based payment resource trading can be achieved using full decentralized/distributed P2P platforms, such as blockchain technology [1,15]. The approach [13] introduced a mechanism known as the Commit to Pay (CTP) to tackle double spending based on blockchain with the help of Merkle Tree [28]. It is a payment agreement that commits a sum of tokens to the seller. Under this process, the buyer will not be able to spend the money deposited, and the deposited money will not be transferred to the producer's account until the products are released else, the token will be reversed back to the buyer after a given time frame. Another untraceable blockchain token-based payment system was implemented in [4]. The technique also explored the idea of Bitmessage technology [29]. Besides, the technique prevents an attacker from spending more than one digital token or ownership of generated commodities. Thus, approaches [4,13] demonstrate unique and better techniques towards curtailing double spending attack in the SMP more than other existing approaches mentioned above. Nevertheless, the approaches [4,13] are only applicable in single resource trading and may not support multiple transactions at a time in a multi-party SMP. Furthermore, they may be prone to Blockchain analysis attack and the implementation of Merkle Tree in a Blockchain-based network such as Ethereum network may be computationally cost which can also lead to slow flow of transactions. This might make the process to become less efficient and less user-friendly. However, based on contextual and practical similarities, [4,13] are set as the benchmark techniques to our proposed approach.

In conclusion, almost all the existing literature only focused on OTOSRT resource trading. These approaches are not well-suited for simultaneous multi-resource trading in a multi-user SMP environment due to the possibility of double spending attack. Thus, there is a need to come up with more scalable lightweight cryptographic solutions that support multiple resource trading. Table 1 provides the summary of the most current related approaches in the literature.

3 System model

This section provides a detailed discussion on our system model, which include; (1) network model, (2) adversary model, (3) security and operational requirements, and (4) requirements for security failure of our proposed framework.

Table 1 Summary of related approaches

Technique	Objectives	Attacks considered	Achievements	Limitations
ECC, ECDSA Blockchain [4]	Token-based payment system	Double spending, Identity and data transaction disclosure	Prevents an attacker from spending more than one digital token or ownership of generated commodities	Limited to a single commodity trading, Parallel trading not possible
Merkle tree, Blockchain [13]	Payment agreement that commits a sum of tokens to the seller	Double spending and Identity disclosure	Prevents an attacker from spending more than one digital token or ownership of generated commodities.	Parallel trading not possible, Limited to a single commodity trading
Blockchain [16–18]	P2P trading and payment	Double spending, 51% attack	On-line manager to confirm tokens before vendors could provide their services to prevent double-spend	Central point of failure, parallel trading not possible, limited to a single resource trading
Blockchain [19,20]	Digital tokens as a payment mechanism for P2P trading	Double spending	Tokens are exchangeable between peers	Tokens are not anonymous, and peers depends on central authority
Blockchain, DHT routing layer [21,22]	Using digital tokens in P2P trading and achieving peer anonymity	Double spending	Real-time double-spending detection using P2P network as a centralized database	Databases cannot be completely trusted especially when peers are compromised
Blockchain [1,3,24–26]	Anonymous P2P transactions using digital tokens	Double spending, identity linking attacks	Tokens are anonymous, peers autonomous, and real-time double spending detection	Prone to double spending especially when nodes are compromised

3.1 Network model

In this work, the SMP (inspired by [30]) comprises of smart homes, equipped with smart gateway hardware (e.g. smart meters) that are connected to the internet, as shown in Fig. 1. Legitimate smart homes in the SMP are identified by their respective smart gateways and can take part in the P2P resource trading either as a resource provider (Seller) or resource requester (Buyer). They initially purchase their individual resources from their respective resource provider (RP) for example internet service provider, water supply company, and energy grid. Later on, participants (Sellers) sell their excess resources to other participants (Buyers) who need extra resources in the community. It is presumed that both the buyer and seller of a given resource must be subscribers of the same RP.

The SMP network is based on semi-centralized private Ethereum blockchain technology with a self-triggering program called smart contract [9,10,14]. Besides, the SMP is a private digital trading environment were all participating stakeholders perform trading activities based on proof-of-authority (PoA) consensus algorithm [6]. In PoA consensus algorithm, a miner node is selected based on its identity; and authority (power) is given to it to verify and allow any transaction to be added to the block. The SMP as a whole, is equipped with a temper-proofed virtual management point referred here as the Smart Market Gateway (SmGW), as shown in Fig. 1. Both the smart homes and SmGW have access to smart contracts and use separate Ethereum Account (EA). The SMP is presumed to have a set of registered and recognized RPs and smart homes in the network is the subscriber of the RP. The RPs provide the needed resources (such as water, electricity, and internet bandwidth) to all participating smart homes. The model is inspired and based on the Bitmessage method of decentralized P2P message authentication and delivery.

3.2 Adversary model

We make the following preliminary assumptions about adversary:

- an adversary may control the means of communication between participants in SMP and then launch double spending attack,
- the SmGW and other smart gateways are temper proof and therefore cannot be compromised.

Motivated by [14], the concept of an attack in this work, is justified on the basis of the following parameters; (i) adversary's likelihood of making a double spending attack (DS), (ii) a potential progress function (P) and (iii) a catch-up function (C). The parameter DS relies on P and C; and intuitively tests the network's susceptibility to a double spending attack. P describes the anticipated adversary branch length after a valid branch has been sufficiently long. While C describes a double-spending attack likelihood by considering the anticipated length of the adversary's branch. Following are the definitions of the parameters that are used to describe the adversary model:

- $q \in [0, 1]$: is the probability that an adversary will be able to mine a block faster than the trustworthy nodes once they start to mine simultaneously. This implies

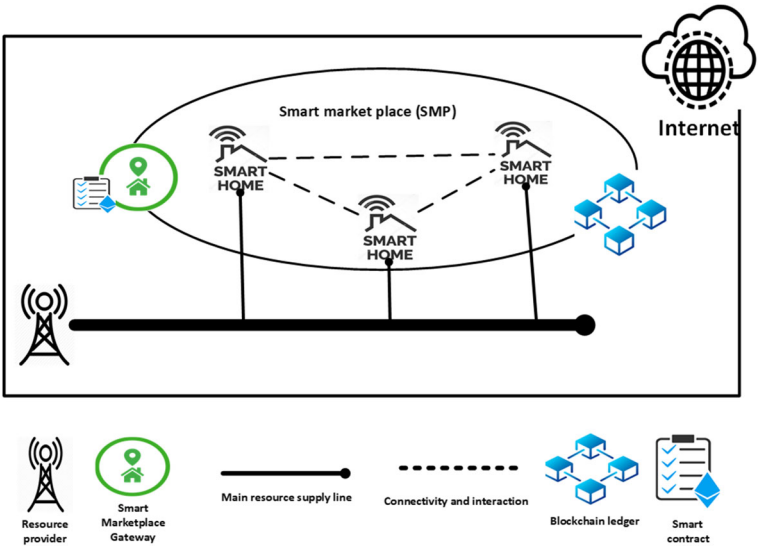


Fig. 1 Proposed system model

that q is the ratio of the adversary's computing strength compared to total network computing capacity.

- The minimum number of verified block confirmations and transactions needed for acceptance set by each seller or buyer (as the case may be) is given as: K.
- The average amount of time in seconds required to mine a block for the entire network (both for trustworthy and adversary nodes) is given as: $\tau \in \mathbb{R} > 0$

Main goal of the model is to analyze the output of parameter DS in relation to other parameters as follows:

- $DS(q, K, n, t)$ is the likelihood that the adversary will effectively execute a double-spending attack, provided that the adversary node dominates q percent of the system, given an initial benefit of n blocks and t seconds over trustworthy nodes, and only the K^{th} block has been mined.
- $P(q, m, n, t)$ is the likelihood that the adversary would precisely mine n blocks after the mining of an m^{th} block by trustworthy nodes. While, t is the time at which an adversary node mines the n^{th} block precisely.
- $C(q, t)$ is the probability that an adversary's branch will be longer than a trustworthy one, if trustworthy nodes were to mine their final blocks t seconds earlier than adversary.

In our attack scheme, states are recognized via the size of legitimate and bogus branches, that are presumed to have the same size, as well as the time gap during which all trustworthy and adversary nodes have mined their last block. In other words, a state comprises of two quantities t and n indicating the time gap t during which both the trustworthy and the adversary nodes mined their n^{th} blocks. Hence, the possibility of double spending attack occurrence in this work is measured based on the ability of

the adversary to mine a block of transactions faster than the trustworthy nodes within a specified practical time advantage as suggested in [14].

3.3 Security and operational requirements

The proposed framework is intended to satisfy the following security and operational criteria:

- *Authentication* The proposed framework should allow only verified-registered and authorized smart homes of the SMP to participate in the resource trading.
- *Double-spending* It should be able to provide protection from the two identified double spending attacks. These include; Token based double spending and Ownership based double spending.
- *Multi-party SMP* The system should be able to allow a participant to trade with other multiple participants in the SMP simultaneously.
- *Multi-resource trading* The framework should allow a participant to trade multiple resources concurrently.

4 Secure multi-resource trading model (SMRT)

This section presents the proposed SMRT trading model as presented in Fig. 2. Four main actors involved in this model are: the requester gateway (*ReqGW*), provider gateway (*ProGW*), the smart market gateway (*SmGW*), and the RP. Others include the auction board, the blockchain handler and the payment handler smart contract (PHSC). The scheme is proposed to be implemented in three phases, namely: registration, initialization and message transmission.

4.1 Registration of smart homes in the SMP

The *SmGW* registers all participating smart homes, and add them to the SMP. In this respect, the joining smart home will need to send a joining request transaction (containing its credentials) to the *SmGW*, which comprises of the smart gateway's public key, RPIDN (resource provider identification number) and block Timestamp. The *SmGW* will verify if the RPIDN belongs to valid RP in the SMP and then evaluate the freshness of the Timestamp as described in Eq. (1). The Timestamp freshness verification is done by checking if the current timestamp is greater than that of the received transaction message. If the results are true, then the smart home is added to the SMP and a reply with a new shared key called Transaction session key (*TSKey*) encrypted with the public key of the smart home is transmitted. After receiving this key, the smart home gateway using its public and private keys creates new pairs of session IDs called marketing session IDs ($MSID_{pub}$, $MSID_{prv}$). All keys and IDs are for one-time usage. Therefore, for subsequent trading cycles, new keys and IDs are generated using public and private keys.

Each pair of Marketing session IDs are created using the Bitmessage technique, which allow every participating node (smart home) to create IDs and broadcast

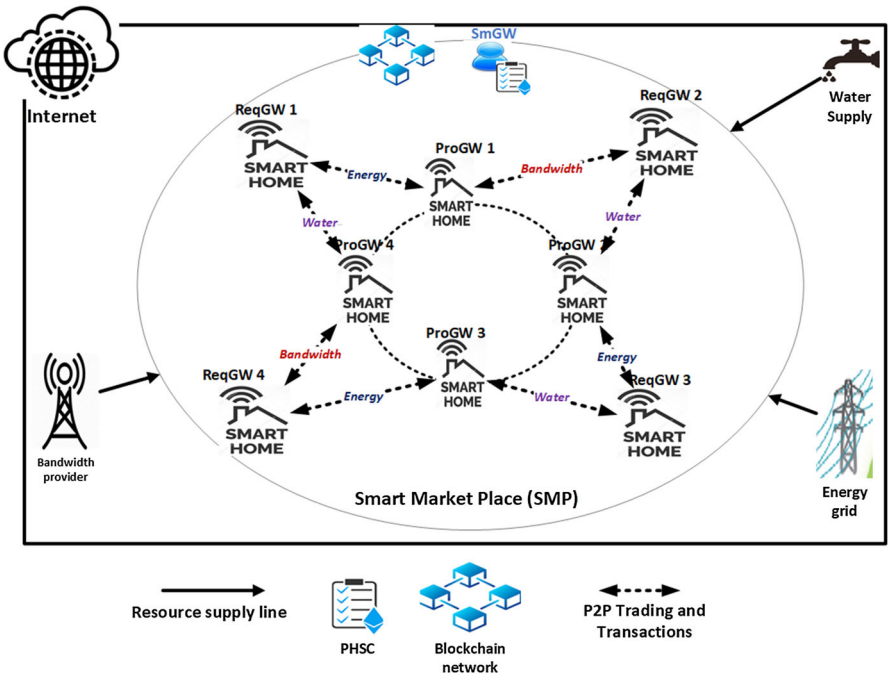


Fig. 2 Proposed trading model

encrypted transaction messages anonymously. However, the Bitmessage technique encrypts each block of transactions independently instead of using an encryption method which mostly makes the contents of the encrypted blocks dependent on each other. With this Bitmessage feature, the attacker can rearrange blocks of the encrypted transaction messages which the receiver will see as a valid message, allowing blocks of the message to arrive in the order chosen by the attacker [31]. To address the message arrival problem, we adopt cryptographic proof-of-authority (PoA) and speedup RSA encryption/decryption [32] schemes in the system during implementation to ensure security and privacy are maintained within the private network.

$$curr.Timestamp > Timestamp \tag{1}$$

4.2 Initialization

The initialization steps required for a transaction among the trading partners (*ProGW* and *ReqGW*) are discussed in this section. Following the smart homes registration in the SMP, the potential *ProGW* submit its tender of resources $T(R)$ and its $MSID_{ProGW_{Pub}}$ first to the respective RPs for resource ownership key generation. The $T(R)$ is given as $\{R_1, R_2, R_3, \ldots, R_n\}$, where n is the cardinality of the resource types, if the *ProGW* have subscriptions with n RPs and willing to make trading of n resources in the SMP

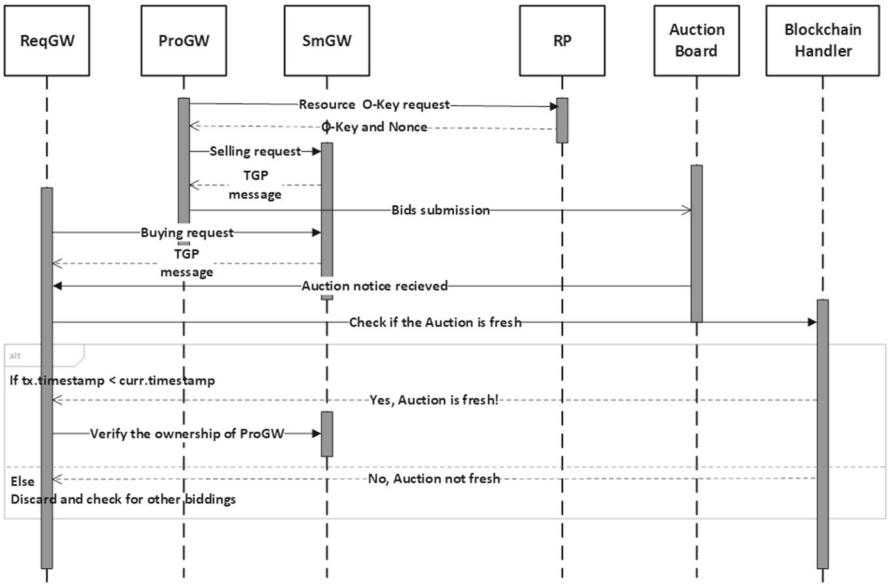


Fig. 3 Initializing the trading process

as described in Eq. (2).

$$\begin{aligned} T(R_1) &= ap_1, r_{R_1}, bp_2, r_{R_1}, cp_3, r_{R_1}, \ldots, zp_x, r_{R_1} \\ T(R_2) &= ap_1, r_{R_2}, bp_2, r_{R_2}, cp_3, r_{R_2}, \ldots, zp_x, r_{R_2} \\ T(R_3) &= ap_1, r_{R_3}, bp_2, r_{R_3}, cp_3, r_{R_3}, \ldots, zp_x, r_{R_3} \\ &\vdots \\ T(R_n) &= ap_1, r_{R_n}, bp_2, r_{R_n}, cp_3, r_{R_n}, \ldots, zp_x, r_{R_n} \end{aligned} \tag{2}$$

Where a, b, c, and z are positive real numbers representing the volumes (amounts) of resource type needed, p and r are the price and unit of the resource types $R_1, R_2, R_3, \ldots, R_n$ respectively.

Then each RP verify the request and the availability of the said resource. Each RP then replies with a secrete ownership key ($OKey_{ProGw}$), a nonce, and the block timestamp to the *ProGW*. Besides, a lock is also activated by the concerned RP on the said amount of the requested resource; this ensures that the resource cannot be used by *ProGW*'s smart home after the ownership key is generated. The RP reply is described in Eq. (3) and line 10 and 19 of Algorithm 1.

$$\begin{aligned} &Tx_{(ProGW)\leftarrow RP} \\ &\quad = \{(OKey_{ProGw_{R_1}}, Nonce_{R_1}), \ldots, (OKey_{ProGw_{R_n}}, Nonce_{R_n})\} \end{aligned} \tag{3}$$

Secondly, on receiving the ownership key(s), the *ProGW* submit a selling participation request to the *SmGW* which is made up of its $MSID_{ProGW_{Pub}}$, $T(R)$ and the

block timestamp encrypted with its *TSKey*. In return, the *SmGW* receives and decrypt the message, evaluate it as described in Eq. (1) and then updates the blockchain public ledger. Also, during this evaluation, the *SmGW* also verifies the validity of the submitted $T(R)$ from the concerned RP. If the evaluation results turned to be true, the *SmGW* replies with a trade granting permission (TGP) message as depicted in Fig. 3. On receiving the TGP message, the *ProGW* broadcast its bids anonymously on the blockchain auction board as given in Eq. (4), which also includes the price of the resource to be sold. The broadcast processes are also depicted as an output of Algorithm 1.

$$\begin{aligned} Tx_{auctionboard \leftarrow ProGW} = MSID_{ProGw_{Pub}} \parallel Type \parallel \\ Volume \parallel Price \parallel Timestamp \parallel Nonce \parallel Unit \end{aligned} \tag{4}$$

The pair of ID, Nonce and $OKey_{ProGw}$ together are used only for one-time trading cycle. Similarly, the pair of IDs and $OKey_{ProGw}$ are also used to prevent double spending from the side of *ProGW*. While nonce is used to verify the identity and ownership claim of *ProGW* on the said resources. In addition, the $OKey_{ProGw}$ is used to sell or transfer ownership of the resource locked by this secret key to the buyer. Bids are broadcasted to all nodes in the private blockchain network.

The *ReqGW*, being the potential resource buyer submits a buying participation request to the *SmGW* which is made up of its $MSID_{ReqGw_{Pub}}$, and the block timestamp encrypted with its *TSKey* as described in line 2 of Algorithm 2. In return, the *SmGW* receives and decrypt the message, evaluate it as described in Eq. (1) and then updates the blockchain public ledger. If the evaluation result turn out to be true, *SmGW* replies with a new trade granting permission (TGP) message. This will enable the *ReGW* to receive broadcasted bids from the auction board. The initialization processes are highlighted in Fig. 3.

4.3 Transactional message block (TMB) transmission

A *ReqGW* obtains a given *ProGW*'s *MSID* and other information about the bid from the auction board. Since there may be several suppliers, therefore the *ReqGW* filters result by activating a matching process according to price and amount. To do so, the auction handler searches the blockchain for recent transactions aimed at a given public key by verifying the freshness of the Timestamp as described in Eq. (1) and line 5–13 of Algorithm 2. The *ReqGW* then obtain the records of active bids and messaging stream Marketing Session IDs of supplier relevant to its query.

If the *ReqGW* selects the *ProGW* as a potential supplier, it sends a private message to *SmGW* asking to verify the *ProGW*'s claim of ownership. This is to re-ensure that authenticity and legitimacy are maintained at this level. From Fig. 4, The *SmGW* scans blockchain public ledger and responds if the argument is true or false (Line 4–13, Algorithm 2). If the claim is true, then the auction panel is utilized for bid selection and private messages are sent to negotiate with a potential supplier (*ProGW*) by encrypting the *TMB* using the $MSID_{ProGW_{Pub}}$ as:

$$MSID_{ProGW_{Pub}}(MSID_{ReqGW_{Pub}}, \omega_i, T_i) = \phi_i \tag{5}$$

Algorithm 1: : Provider Gateway (ProGW) resource submission and initialization

Input: Read the value: $MSID_{pub_{ProGW}} \leftarrow hash(PubKey_{ProGw}, Timestamp2)$
Output:
$Broadcast Bidding \leftarrow MSID_{Pub_{ProGW}} \| Type_i \| Volume_i \| Price_i \| Timestamp_i \| Nonce_i \| Unit_i$

```
1  Function SubmitTender (T(R)):
2      n ←Number of resources
3      if (n = 1) then
4          R ← Type of resource;
5          a ← Amount of resource;
6          p ← Price of resource;
7          r ← Unit of resource;
8          T(R) ← (a, p, r);
9          Tender.Submit ←(T(R), MSID_{Pub_{ProGW}}, Timestamp);
10         RP.Reply ← (OKey_{ProGw}, Nonce).
11     else
12         for (i = 0; i <= n; i + +) do
13             R_i ← Type of resource_i;
14             b_i ← Amount of resource_i;
15             p_i ← Price of resource_i;
16             r_i ← Unit of resource_i;
17             T(R_i) ← (b_i, p_i, r_i);
18             Tender.Submit ←(T(R_i), MSID_{Pub_{ProGW}}, Timestamp);
19             RP.Reply ← (OKey_{ProGw}, Nonce).
20         end
21     end
```

where ω_i is the request message and T_i is the block timestamp. The *ReqGW* identity remains anonymous behind a pseudonym $MSID_{ReqGW_{Pub}}$.

A validating authority validates the *TMB* and adds it to the blockchain based on PoA consensus algorithm. With this, the encrypted TMB is published in the blockchain network where all participant nodes will receive a copy of the package. Only *ProGW* which has the right secret key will be able to decrypt the *TMB* using a $MSID_{ProGw_{Prv}}$, as follows:

$$MSID_{ProGw_{Prv}} \phi_i = \omega_i, T_i, MSID_{ReqGw_{Pub}} \tag{6}$$

and then *ProGW* verifies the freshness of T_i using Eq. (1). If it verifies to be true, then some response should be returned to the *ReqGW* by encrypting another *TMB*:

$$MSID_{ReqGw_{Pub}} \omega, T_j = \omega_j \tag{7}$$

Similar checks will be performed by *ReqGW* to verify the message sent by *ProGW*. As depicted in Fig. 4 and lines 15–21 of Algorithm 2, after all sides agreed on the negotiation, then *ReqGW* trigger a payment handler smart contract (PHSC) and then submit the payment via the PHSC in the form of encrypted anonymous message stream transaction as demonstrated in the Eq. (8). A commit to pay notification transaction will be broadcasted by PHSC to inform the nodes about the commitment of the *ReqGW*

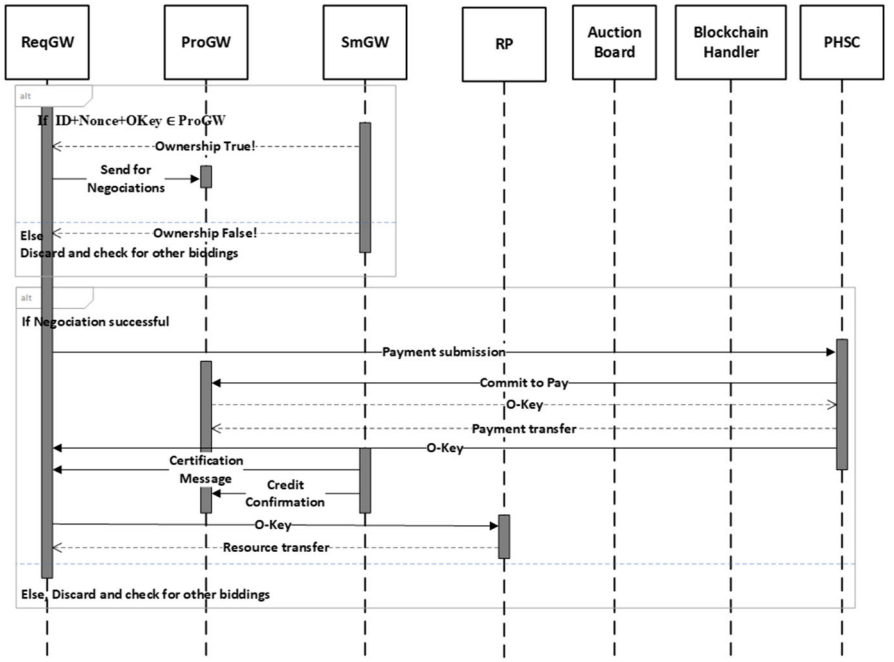


Fig. 4 Trading process conclusion

as depicted in Eq. (9).

$$Tx_{ProGW \leftarrow ReqGW} = MSID_{ProGw_{Pub}}(MSID_{ReqGw_{Pub}}, Type, Volume, Unit, Token, T_i) \tag{8}$$

$$Tx_{(ProGW) \leftarrow PHSC} = T_{ID} \parallel Timestamp \parallel t' \parallel Price \parallel ContractHash \parallel MSID_{ReqGw_{Pub}} \tag{9}$$

Where T_{ID} is the transaction ID, t' is the maximum waiting time, *Price* is the agreed price of the commodity, *ContractHash* is the hash of the smart contract, $MSID_{ReqGw_{Pub}}$ is the public pair of the *ReqGW*'s marketing session ID and *Timestamp* is the block timestamp.

This will enable the *ProGW* to re-confirm its willingness to sell its *Okey* of the said resource before the PHSC could allow the payment to reach the respective *ProGW*. As per our PHSC rule, payment can only be released when the PHSC confirms the receipt of the appropriate *Okey*. The same rule applies before *Okey* could be allowed to reach the respective *ReqGW*.

After seeing the commit payment notification from the PHSC, the *ProGW* then encrypts its $OKey_{ProGw}$ and submit it to the *ReqGW* anonymously via the PHSC within the waiting time t:

$$Tx = MSID_{ReqGw_{Pub}}(OKey_{ProGw}) \tag{10}$$

Algorithm 2: : Requester gateway buying transactions and negotiations

Input: Read the value: $MSID_{pub_{ReqGW}} \leftarrow hash(PubKey_{ReqGW}, Timestamp4)$
Output: $OKey_{ProGW}$, $Payment$

Function `BuyingTransact(BT)`:
  $buyingRequest \leftarrow hash(MSID_{pub_{ReqGW}}, Timestamp)$
  $ReqGW.LocalRepository \leftarrow BroadcastBidding$
Function `Auction_Check(AC)`:
  if $(curr.Timestamp > Timestamp)$ **then**
    if $Nonce_i$, ID_i, $OKey_i \in ProGW_i$ **then**
      $Return$: Send for Negotiation
    else
      $Return$: Claim not Valid!
    end
  else
    $Return$: Auction freshness not confirmed!
  end
Function `Negotiations(NG)`:
  if $(Price_i = Accept)$ **then**
    $Return$: Negotiation successful!
    Trigger $PHSC$
    $PHSC \leftarrow ReqGW.Payment$
    $ProGW \leftarrow PHSC.CommitPay$
    $PHSC \leftarrow ProGW.OKey_{ProGW}$
    $ProGW \leftarrow Payment$
    $ReqGW \leftarrow OKey_{ProGW}$
    .....
    $ProGW \leftarrow SmGW.CreditConfirm$
    $ReqGW \leftarrow SmGW.Certificate$
  else
    $Return$: Auction negotiation discarded!
  end

The PHSC credits the *ProGW*'s account with the said Tokens and then prompts the *SmGW* to sends two messages. The first message is a certification message submitted to *ReqGW*. The message includes the $OKey_{ProGW}$ and *ContractHash* encrypted with the *ReqGW*'s shared *TSKey*. The second message is credit confirmation message which includes the amount of the Token credited, and the *ContractHash*. The message is submitted to *ProGW* encrypted with *ProGW*'s shared *TSKey*.

However, the *ProGW* can simply ignore the commit payment notification without replying with its *Okey* if the seller no longer intends to sell the ownership key. Thus, after the period $t \geq t'$ is achieved, then the PHSC will refund the *ReqGW* and then prompt the *SmGW* to send a transaction cancellation notification to the *ProGW*. The *SmGW* also sends another message to *ProGW* to confirm if it wants to withdraw its bids from the SMP. If the *ProGW* confirm yes, then all its bids will be dismissed from the auction board and a trading withdrawal notification will be submitted to the *ProGW*. Else, the bids will remain available to the buyers.

5 Security analysis

The goal of this section is to demonstrate that the proposed model is protected under the given adversary model. Theorems 1 and 3 are for ownership based double-spending, while Theorem 2 is applied to token based double spending. Similarly, the security analysis is proven to be effective under practical time advantage and average computing power. This is to say that, the solution is effective within the assumed time benefit given to an adversary that has average processing power over the trustworthy nodes. Below are the main security assessments.

Theorem 1 *The system is immune to a malicious ProGW who intend to use-up or not to deliver resource(s) to the ReqGW after receiving payment.*

Proof After resource *Okey* request submission to the concerned RP, the RP verifies and activates a lock on the said resource amount. This will make the *ProGW* to temporary lost control of the resource as soon as he receives the *Okey* of the said resource. Besides, the payment will be confirmed after the *ProGW* released its *Okey* within a given waiting time t. This is to prevent the malicious *ProGW* from intending to use or not to release the purchased resource after receiving his payment. Furthermore, the *ProGW* loses complete control of the resources as soon as he releases his *Okey* within the given waiting time t and the payment will be confirmed after. The payment is given as:

$$ProGW_{payment} = PHSC(OKey_{ProGW}) \cdot t \leq t' \tag{11}$$

The payment is reversed to the payer and the transaction void if $t \geq t'$, where t' is the maximum waiting time. This is ensured by the PHSC. *ProGW* may withdraw from the market and gain full custodial control of his remaining resources (if applicable) after the system confirms and balances all pending transactions. □

Theorem 2 *A malicious consumer (ReqGW) cannot receive any resources locked by relevant Okey without paying the respective producer (ProGW) nor can he spend digital token more than one time in the system.*

Proof Even if the malicious *ReqGW* intend to compromise the agreed deal by not paying the requested fee, the PHSC will not allow the passage of the *ProGW*'s *Okey* until it confirms the release of the agreed fund from the concerned *ReqGW*. The ownership key release is given as:

$$ReqGW_{release} = PHSC(Token_{ReqGW}) \cdot t \leq t' \tag{12}$$

The order is reversed to the seller and the transaction void if $t \geq t'$. In addition, the proposed model imposes a principle that only previously unused transaction outputs can be used as inputs for new transactions. A validating authority node verifies this principle while disseminating transactions by traversing and tracking the whole of their local blockchain copy. □

Theorem 3 *The system can prevent attacker to spend ownership over submitted tender of resources more than one time within practical time advantage and average computing power.*

Proof To prevent double spending on resource tender $T(R)$ ownership, the RP "locks" $T(R)$ with a unique ownership secret key $OKey_{ProGw}$, and reply with a copy of the key to the *ProGW*. These processes are carried out after the tender is received by the RP. However, when the *ProGW* submit a selling request, the *SmGW* contact the concerned RP for verification of the *ProGW*'s claimed $T(R)$. The RP verifies the claim and replies to *SmGW* with verification results accordingly.

If the $T(R)$ is legitimate and not being used, the *SmGW* replies with a TGP message to the potential *ProGW* and it will be allowed to submit its bids in the auction board. Otherwise, it will be denied access and the $T(R)$ will be discarded. These processes are used to prove $ProGW_j$'s identity and ownership over $T(R_j)$. Unique values (*Okey* and *Nonce*) also prevents adversaries from spoofing $ProGW_j$'s identity. □

6 Testing and evaluation

This section presents an experimental analysis of our model. The resilience of the proposed model is tested and assessed against double spending attacks on the basis of partial progress towards block production, which is driven by time advantage and average computing power. In addition, a simulation-based analysis and comparison of the proposed model was carried out to determine the level of performance of our smart contract based on its computational overhead and latency of transactions compared to other existing models. The smart contract of our proposed scheme was deployed on the virtual blockchain. The contract deployment transaction details is also available in Etherscan: https://rinkeby.etherscan.io, and the deployment results/details is provided in Fig. 5 which include the gas consumption and transaction cost of the contract deployment.

The proposed model was implemented using Solidity Language. Experiments were conducted using the Metamask [33] Ethereum wallet and chrome plugin. The proposed model is design on the basis of PoA consensus algorithm, as a result PoA based Rinkeby testnet [34] was used to alternatively mimic the blockchains ledger and the testing network. Using this testnet, the real-world situation can be replicated efficiently. Performance comparison of the proposed solution is provided against Aitzhan et al. [4] and Dorri et al. [13], and they were implemented using the same setup scenario to have better observations and readings of the experiment.

During the implementation of the smart contract, the smart gateways are represented as mappings from the Ethereum address of the nodes to objects containing information about the participating entities. The Ethereum address of the contract owner (manager) was initialized during smart contract deployment. The owner holds the highest authority (private network administration) and has access to all features of the framework. Furthermore, we registered and mapped thirty (30) different Ethereum addresses to characterize smart home gateways (*ProGW* and *ReqGW*) in the system, which are later allowed to interact with each other in the *SMP*. MetaMask is switched to the appropriate Ethereum account each time *ProGW* wants to make tender submission or *ReqGW* wants to perform trading. Events with enough information were conducted to observe the performance of each process. Several scenarios had been checked to verify the logic of the smart contract code. The proposed smart

Fig. 5 Smart contract deployment results

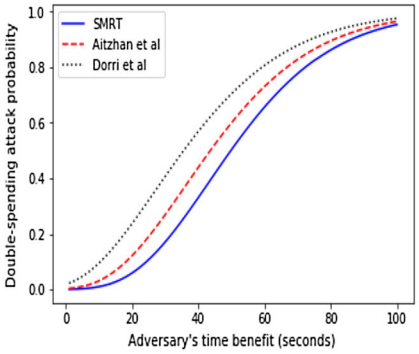
contract was analyzed using the Oyente smart contract security analysis tool [35]. Results show that the smart contract is free from established security vulnerabilities [36] such Reentrancy, Timestamp dependence, Transaction dependency, and Parity multisig bug with 53% EVM Code Coverage. The codes of this work is available in GitHub repository (https://github.com/sysbel07/Multi-Commodity-Trading-scheme-with-many-smart-marketplace-SMP-participants) for public usage.

The following subsections provide performance evaluation of the proposed model in terms of resilience against double spending attack, computational overhead and transaction latency. The double spending attack evaluation process involves testing the system's likelihood towards the attack. The evaluation was based on partial progress towards block production in the blockchain [14]. The computational overhead evaluation is based on execution gas and transaction gas consumed by the system during function call operations. The execution gas is defined as the metric for measuring the cost for smart contract implementation. Whereas the transaction gas includes the cost of implementing and adding the contract to the blockchain. The transaction latency is another parameter used in the evaluation, it is usually described as the measure of how long transaction messages take. The parameters are based on the average or percentile of the latency over a sample number of transaction messages [37].

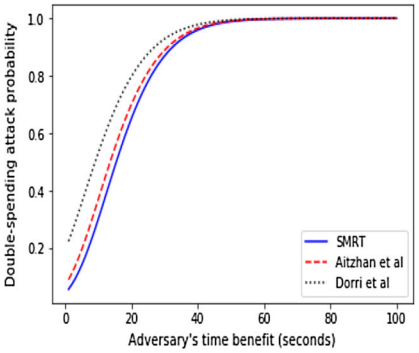
6.1 Double spending attack mitigation evaluation

This section outlines experimental findings obtained following the launch of token and ownership based double spending attacks on the proposed SMP network. The experiment was implemented using Python 3 programming language. The experimental results presented in this section are compared against the benchmark techniques. The recorded number of blocks during our *SMP* implementations and their generation times where used as our inputs to this experiment.

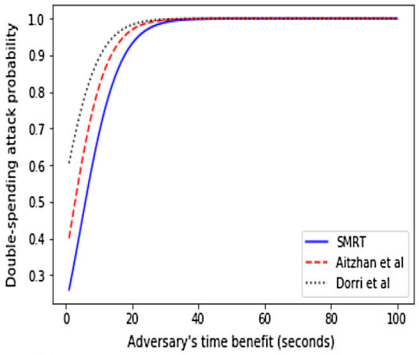
Similar arbitrary time advantages (from 0 to 100 s) were used for all the test subjects. Our results show that the likelihood of a double-spending attack increases with an increase in the time benefit or the power of the adversary. Also, it can be observed from Fig. 6a–e that double-spending attacks are very rare if there is a limited time



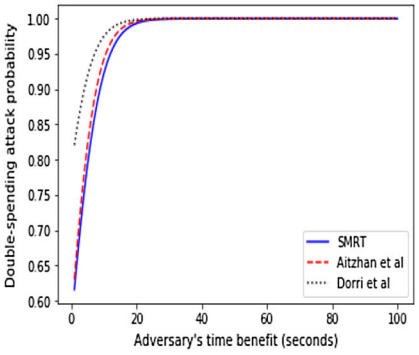
(a) blockchain-based double spending probability based on the time benefit, given q=10%.



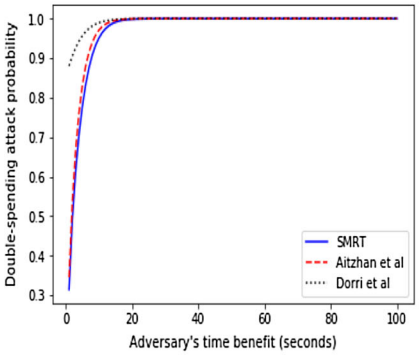
(b) blockchain-based double spending probability based on the time benefit, given q=20%.



(c) blockchain-based double spending probability based on the time benefit, given q=30%.

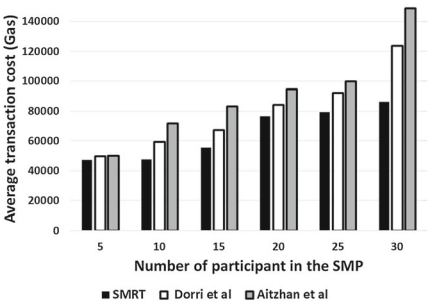


(d) blockchain-based double spending probability based on the time benefit, given q=40%.

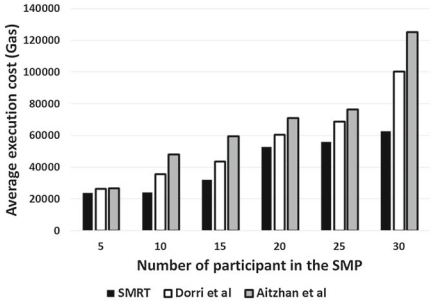


(e) blockchain-based double spending probability based on the time benefit, given q=50%.

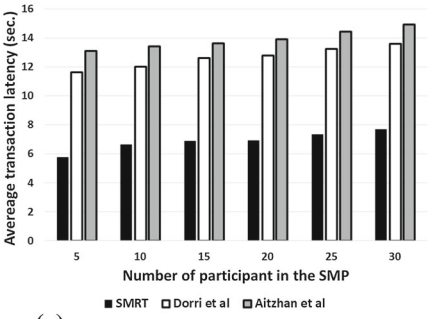
Fig. 6 Double spending attack mitigation evaluation



(a) Average transaction cost of the frameworks.



(b) Average execution cost of the frameworks.



(c) Average transaction latency of the frameworks.

Fig. 7 Computational overhead and transaction latency. **a** Average transaction cost of the frameworks. **b** Average execution cost of the frameworks. **c** Average transaction latency of the frameworks

advantage to the adversary. The above situations might be as a result of representing the possibility of a double spending attack as the likelihood of a time disadvantage of t seconds before a $K+1$ block is mined multiply by the likelihood of keeping up with that disadvantage.

However, from Fig. 6d and e, our proposed model proves to be promising even with 40–50% of the computing power belongs to the adversary. This is due to the various non-repeated sessional IDs used based on Bitmessage techniques and the ability of our PHSC to track every single token spent during each transaction. Moreover, to provide a detailed understanding of how the double spending behaves, the graphs are presented under five separate scenarios, namely $q = 10\%, 20\%, 30\%, 40\%$ and 50% as shown in Fig. 6a–e respectively.

The experiment revealed that both [4,13] are not completely successful in addressing double spending attacks (Fig. 6a–e). The major advantage of the proposed SMRT model is that it provides a OTMMRT facility and also manage to reduce the risk of double spending attacks.

6.2 Computational overhead and transaction latency

Practically, once an Ethereum blockchain transaction is made, Ethers are charged as gas. In Ethereum, the computational overhead is measured based on the amount of gas consumption by a system [37]. In this model, we simulated several potential *OTMMRT* scenarios, and compared the results of the proposed SMRT model with the benchmark models [4,13]. The transaction and execution gas consumption are shown in Fig. 7a and b for the overall system operations, respectively. These results show that gas consumption in SMRT is least effected by the increase in the number of participants in SMP compared to existing schemes. It is because we avoid the use of clustering algorithm that involves more computations in the development of smart contracts; since using these practice creates more and unnecessary gas consumption in the system [37].

Additionally, the transaction latency of our proposed system is recorded, measured, and compared to that of benchmark works. Transaction latency in our approach was measured using a background timer application that runs during transaction execution. Time is measured based on the system processor clock and depends on the current processor schedule. It is observed from Fig. 7c that our proposed model shows less transaction latency as the number of participants increases in the *SMP* as compared to [4,13]. This is consistent with the fact that the overall block time for validation is less than 8 s in our framework according to Etherscan. While validation of a message block in both [4,13] takes approximately up to 15 s. In addition, techniques [4,13] appear to be more pronounced than our approach. This was because the techniques drive their prevalence by depending on a central or semi-centric controller using PoW consensus mechanism, reuse of keys/IDs and broad encryption/decryption processes. While in our proposed approach, the PoA chains are maintained by trusted parties, keys/IDs are sessional and the process uses a speed-up encryption/decryption mechanism.

7 Conclusion

The study proposed a blockchain-based semi-decentralized multi-resource trading model named SMRT. Multiple scenarios were created to gauge the feasibility of multiple parallel trading interactions. An adversary model was developed to quantify the effect of double spending attack on the proposed model. Our findings indicate that double spending attack becomes a significant risk once an adversary has ample time advantage towards bogus block production with adequate computing power. Besides, SMRT demonstrates to be more promising in OTMMRT trading scenarios and at the same time deals with double spending attack at a low computational cost and has minimal transaction latency as compared to the benchmark techniques. Compared to the existing trading models, it has been noted that SMRT not only provides protection against double spending attacks, but also offers a reduction of up to 50% in its computational overhead.

Due to the transparency of blockchain, participants may be subject to the disclosure of confidential information, such as the identity of the participant and details of transactions carried out during trading process. As a result, ensuring the privacy of

participants becomes a challenging task in most blockchain-based trading models. In the future, we intend to propose a model to address issues related to privacy of the participant in a blockchain-based trading model.

References

1. Gai K, Wu Y, Zhu L, Qiu M, Shen M (2019) Privacy-preserving energy trading using consortium blockchain in smart grid. IEEE Trans Ind Inform 15:3548–3558
2. Krishnamachari B, Power J, Kim SH, Shahabi C (2018) I3: An IoT marketplace for smart communities. pp 9–10
3. Lin CC, Deng DJ, Kuo CC, Liang YL (2018) Optimal charging control of energy storage and electric vehicle of an individual in the internet of energy with energy trading. IEEE Trans Ind Inform 14(6):2570–2578
4. Aitzhan NZ, Svetinovic D (2018) Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. IEEE Trans Depend Secur Comput 15(5):840–852
5. Tushar W et al (2019) A motivational game-theoretic approach for peer-to-peer energy trading in the smart grid. Appl Energy 243:10–20
6. Alam MR, St-Hilaire M, Kunz T (2019) Peer-to-peer energy trading among smart homes. Appl Energy 238(January):1434–1443
7. Tushar W et al (2015) Three-party energy management with distributed energy resources in smart grid. IEEE Trans Ind Electron 62(4):2487–2498
8. Tushar W, Yuen C, Mohsenian-Rad H, Saha T, Poor HV, Wood KL (2018) Transforming energy networks via peer-to-peer energy trading: the potential of game-theoretic approaches. IEEE Signal Process Mag 35(4):90–111
9. Reyna A, Martín C, Chen J, Soler E, Díaz M (2018) On blockchain and its integration with IoT. Challenges and opportunities. Future Gener Comput Syst 88(2018):173–190
10. Banerjee M, Lee J, Choo KKR (2018) A blockchain future for internet of things security: a position paper. Digit Commun Netw 4(3):149–160
11. Paris S, Martisnon F, Filippini I, Clien L (2013) A bandwidth trading marketplace for mobile data offloading. In: Proceedings—IEEE INFOCOM, pp 430–434
12. Bhatia R, Chuzhoy J, Freund A, Naor JS (2007) Algorithmic aspects of bandwidth trading. ACM Trans Algorithms 3(1):10
13. Dorri A, Luo F, Kanhere SS, Jurdak R, Dong ZY (2019) SPB: a secure private blockchain-based solution for distributed energy trading. IEEE Commun Mag 57(7):120–126
14. Pinzón C, Rocha C (2016) Double-spend attack models with time advantange for bitcoin. Electron Notes Theor Comput Sci 329:79–103
15. Sousa T, Soares T, Pinson P, Moret F, Baroche T, Sorin E (2019) Peer-to-peer and community-based markets: a comprehensive review. Renew Sustain Energy Rev 104:367–378
16. Alcarria R, Bordel B, Robles T, Martín D, Manso-Callejo MÁ (2018) A blockchain-based authorization system for trustworthy resource monitoring and trading in smart communities. Sensors 18:3561
17. Russo C (2018) DENT: a blockchain marketplace for unused mobile data. https://sludgefeed.com/dent-tokenizing-the-mobile-data-industry/. Accessed 29 Apr 2019
18. Makhdoom I, Zhou I, Abolhasan M, Lipman J, Ni W (2020) PrivySharing: a blockchain-based framework for privacy-preserving and secure data sharing in smart cities. Comput Secur 88:101653
19. Yang B, Garcia-Molina H (2003) PPay: micropayments for peer-to-peer systems. In: Proceedings of the ACM conference on computer and communications security, section no. 3, pp 300–310
20. Xu R, Chen Y, Blasch E, Chen G (2018) BlendCAC: a blockchain-enabled decentralized capability-based access control for IoTs. pp 1–27
21. Wei K, Chen YFR, Smith AJ, Vo B (2006) WhoPay: a scalable and anonymous payment system for peer-to-peer environments. In: Proceedings of the international conference on distributed computing systems
22. Zhang L, Zhao L, Yin S, Chi CH, Liu R, Zhang Y (2019) A lightweight authentication scheme with privacy protection for smart grid communications. Futurr Gener Comput Syst 100:770–778

23. Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H (2001) Chord. In: Proceedings of the ACM SIGCOMM '01 Conference, pp 149–160
24. Hoepman JH (2010) Distributed double spending prevention. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol. 5964 LNCS. Hoepman, Niu, SAggarwal, pp 152–165
25. Niu C, Zheng Z, Wu F, Gao X, Chen G (2019) Achieving data truthfulness and privacy preservation in data markets. IEEE Trans Knowl Data Eng 31(1):105–119
26. Aggarwal S, Chaudhary R, Aujla GS, Jindal A, Dua A, Kumar N (2018) EnergyChain. In: SmartCitiesSecurity'18, June 25, 2018, Los Angeles, CA, USA, no. June, pp 1–6
27. Osipkov I, Vasserman EY, Hopper N, Yongdae K (2007) Combating double-spending using cooperative P2P systems. In: Proceedings of the international conference on distributed computing systems
28. Blockchain C, Merkle tree—necessity or atavism?. Medium. https://medium.com/@credits/merkle-tree-necessity-or-atavism-d8ff3e263131
29. Warren J (2012) Bitmessage: a peer-to-peer message authentication and delivery system. White paper (27 November 2012) https://bitmessage.org/bitmessage.pdf
30. Li X, Lu R, Liang X, Shen X, Chen J, Lin X (2011) Smart community: an internet of things application. IEEE Commun Mag 49(11):68–75
31. Agrawal A (2013) Bitmessage: communication from scratch. Finextra https://www.finextra.com/blogs/fullblog.aspx?blogid=7920. Accessed 06 Oct 2010
32. Paar C, Pelzl J (2013) Understanding cryptography a textbook for students and practitioners
33. Metamask (2019) Brings Ethereum to your browser. Metamask Chrome extention. https://metamask.io/
34. Rinkeby (2019) Transaction details. https://rinkeby.etherscan.io/tx/0xe685f0ea29afce5d5a86265e87416be613dd36878570ddd71e49cd9d6444f263. Accessed 15 May 2019
35. Luu L, Chu DH, Olickel H, Saxena P, Hobor A (2016) Making smart contracts smarter. In: Proceedings of the ACM conference on computer and communications security, vol 24. 28-October-2016, pp 254–269
36. Praitheeshan P, Pan L, Yu J, Liu J, Doss R (2019) Security analysis methods on Ethereum smart contract vulnerabilities: a survey. pp 1–21
37. Wood G (2014) Ethereum: a secure decentralised generalised transaction ledger (yellow paper) 2017. Ethereum Proj. yellow paper

Affiliations

Bello Musa Yakubu[1] · Majid I. Khan[1] · Nadeem Javaid[1] · Abid Khan[2]

✉ Majid I. Khan
majid_iqbal@comsats.edu.pk

Bello Musa Yakubu
bellomyakubu.cui@gmail.com

Nadeem Javaid
nadeemjavaidqau@gmail.com
http://www.njavaid.com

Abid Khan
abk15@aber.ac.uk

1 COMSATS University Islamabad, Islamabad 44000, Pakistan

2 Department of Computer Science, Aberystwyth University, Aberystwyth, Wales SY23 3DB, UK