A Countermeasure for Double Spending Attacks on Blockchain Technology in Smart Grid

Lanqin Sang, Henry Hexmoor
Southern Illinois University at Carbondale

March 31, 2023

,

Abstract

As a distributed technology, blockchain has been applied in many fields. Much research has been done on its inherent security issues. Among these security issues, double spending is one of the most pernicious. Current countermeasures are not systematic, they either focus on monitoring or detection with no effective strategy to prevent future double spending. These countermeasures also have serious drawbacks, such as high network traffic, high CPU utilization, and heavy management overhead. In this paper, we present a systematic approach to address double spending attack on smart grid. A reputable node is selected, which constantly compares all transactions in current time window with previously validated block and current block. Upon discovering conflicting transactions, a warning message with the conflicting transaction and two penalty transactions are broadcasted to the network to stop the current attack and to prevent future attacks. Our experiment has demonstrated our design is highly effective to detect double spending, with short detection time and low CPU utilizations.

Keywords: Double Spending, Smart Grid, Blockchain, Reputation-Based, Consensus.

1 Introduction

As a decentralized, distributed, and immutable ledger technology operating on p2p network [1], blockchain technology has been applied in many fields to improve security in distributed systems, such as Internet of Things (IoT), healthcare, supply chain, financial services [2], and future energy systems [3]. For example, blockchain has been used to perform security and derivative transactions [4], [5], digital payment [6], [7], [8], data sharing in energy management to address energy traders' privacy in smart power grid [9], [10], a novel blockchain-based energy framework to exchange excess energy among neighboring nodes to ensure privacy preservation [11].

Blockchain constructs blocks registering of different distinct transactions. With an internal consensus mechanism, it guides the system to produce accurate and identical information across the entire network. Blockchain technology is designed to overcome security challenges and enhance data integrity. Thus, security plays an important role to guarantee blockchain acceptability. However, the involvement of monetary assets raises security concerns [12]. Double spending, eclipse attacks, selfish attacks, and flash attacks are all common examples [13]. According to [14], Sybil and double spending risks are the utmost concerns in blockchain systems.

Double spending attack is a type of data integrity attack. A double spending attack occurs when an attacker tries to spend the same token or money more than once [15]. In general, double spending is a technique that is used to deceive someone about the state of a transaction [16]. In recent years, the proposed strategies against double spending and selfish mining consist of monitoring, checking, alter forwarding, alter broadcasting, as well as conceptual research proposals [2]. A few countermeasures to double spending have been proposed [18], [19], [20]. These methods largely focus on broadcasting, confirmation, and enforcing listening periods. They also suggest inserting observers into the network and blocking incoming connection requests. Broadcasting would alert the system of an attack and provide the miners with the problematic transactions. Confirmation is the strategy to check the number of blocks where a transaction appears or to inspect propagation depth of transactions. Due to the time that it takes to authenticate a transaction between a vendor and its client, a transaction may be recorded without full confirmation depth review. Although no amount of confirmation depth will be able to completely prevent such attacks, it is an effective measure to mitigate attacks in the system. Implementing a listening period allows neighboring miners or even sensors to be able to spy on and watch over the blockchain [15]. The effect of enforcing listening periods is also limited because the attack can occur after the listening period, though it is helpful to counter double spending attacks. Inserting observers into the network to forward all transactions to the vendor increases the opportunity that the transactions can be detected in a listening period. This method requires managing the observers.

In the previously proposed strategies, the conflicting transactions were not handled, the resources were wasted because all peers were checking double spending transactions, and the monitoring window was short. Since the observers inserted into the network required management overhead, they demand increased network traffic and CPU utilization. The observers may also cause DDoS attack due to its special traffic pattern. These strategies do not explain how conflicting transactions will affect the consensus results and how to prevent the offending nodes to implement double spending attack again. Another issue with blockchain is even if a block passed the consensus, it only means that all the nodes received the same set of transactions. This does not mean that all the transactions in the block are truly accurate, even though all the nodes are honest. This is because there are no mechanisms to test if these transactions are correct from their sources, i.e., one node's evaluation to another node, the money a node needs to pay. Incorrect transactions will cause disputes among users and damage network reputation. In this paper, we propose a design that will address these issues. Our contribution is we use a single reputable node as the detector to check double spending transactions, and our design is a systematic countermeasure, which will monitor, detect, warn, penalize double spending, and prevent it in future. Below are the key points in our solution:

1. The detection results of double spending attacks will be included in the block consensus.
2. One node specifically acts as the attack detector, which frees other nodes to perform other duties. This detector checks the transactions during the whole transaction receiving and consensus time.
3. Only one of the most reputable nodes is selected as the detector, the detector changes frequently, and its communication is not much different than other nodes'. These two features reduce the chance of the node to be the target of DDoS.
4. The attack detector and other nodes work in a parallel fashion. The detection node checks double spending during the whole transaction collecting time interval and sends out its checking results upon finding any conflicting transactions.
5. As penalty, the detector will create two transactions, one requiring the offending node to pay the value in the conflicting transaction to the detector, the other reducing the offending node's reputation scores. Lower reputation scores will reduce the node's chance to participate in important tasks in the future, including creating transactions.

The rest of the paper is organized as follows. Section 2 reviews related work, Section 3 discusses system design, Section 4 presents experimental result analysis, and the last section contains the conclusion.

2 Related works

A secure blockchain network depends on the safety and security of the nodes involved. As more nodes join the blockchain network, the ingenuity of attacks in the chain will progressively become more susceptible. Among all the security attacks, double spending causes the most concern. Double spending usually targets sellers or vendors. A successful attack would be that the money and service are taken by the attacker, leaving the seller with nothing. This would cause honest nodes to hinder functioning due to the lack of security in their transactions. Much research has been conducted to find solutions to mitigate or eliminate double spending attack in blockchain network.

A method that blocks incoming connection requests was proposed [20]. This essentially prevents one kind of double spending that requires the attacker connect to the vendor directly. By blocking incoming connection requests, the attacker cannot establish a direct connection to the vendor to send the vendor the offensive transaction. However, newly joined vendors must request connections to other peers to ensure they have the latest block chain information. The attacker can use this opportunity to create malicious nodes and distribute them throughout the network. The attacker hopes the new vendors would randomly connect to some of these malicious nodes.

A forwarding framework in [21] increases the amount of confirmation to make it harder to attack. Increasing confirmation would require more authentications to be made in the system in order to confirm a transaction. Based on the hash rate of a sender, the amount of confirmation was calculated, which would be adequate to mitigate double spending attacks. The researcher concluded that when their probability methods to combat attacks is applied, if an attacker controls more hash rate than the honest mining network, the success rate of the attack will still be 100%. A forwarding mechanism in [22] uses peer monitoring techniques to alert the nodes in the system that there are attacks on the blockchain. If the nodes configure the alert system to avoid receiving alerts, they will be vulnerable to attacks. A method proposed in [20] requires the vendor to wait for a transaction to propagate a number of steps before accepting it. The idea is that if more nodes have seen the transaction, it is more likely trustworthy and the greater depth is assumed to be better. However, with a chain of malicious nodes, an attacker could simply move offensive transactions along until the propagation reaches the required depth.

A dynamic observation method in [23] proposed the ENHOBS (enhanced observers) method, which used active observers with indistinguishable traffic patterns for valuable transaction inspection. To detect double spending attack on the network, a one-time scan was run on the blockchain to find duplicate transactions. When matching transactions were detected, an alert would be sent through the network. Once the alert was received and was seen as having verifiable proof of an attack, any transactions matching the same input value would be dropped from the memory pool immediately. A method proposed in [19] requires peers to conduct a deeper investigation of conflicting transactions and broadcast alerts to all peers if a double spending attack is detected. This approach can catch double-spenders only after an attack has occurred, and there is no prevention for future occurrence. Even if the attacker was put on a blacklist, the attacker could create a new pseudonym easily and attack again.

A listening period was used in [18] to monitor all transactions that have been previously received and checked if there were attempts to double spend. If there were, an alert would be sent out to the network. This will not be effective in detecting attacks because the attacker can delay sending the attacking transactions until the monitoring window has expired. Another technique proposed in [18] is to randomly insert observers across the P2P network, which forwards all transactions in the monitoring period to help detect double spending because at least one of the observers will receive conflicting transactions, if there are any. If an attack is detected, an alert message will be sent to the network. This approach is somewhat effective. However, it does not directly prevent the double spending attack or the propagation of the offensive transaction. Plus, the observer's traffic patterns can be easily analyzed by an attacker [24], who can carry out DDoS attacks against the observers and re-enable double spending.

A broadcasting programming strategy in [25] proposed a mechanism to construct special transaction outputs to combat double spending. The output of a bitcoin transaction includes two fields: the first one indicates the amount of bitcoins that will be deposited, the second field, named FR-P2PK (fixed-r-pay-to-pubkey), defines the conditions under which this output could be spent. Such output can be spent with a single signature but has the property that if two different signatures have the same output, which indicates a double spending attack, the private key used to sign the transaction is revealed. Then the observer can generate a third transaction spending the same output and send the amount to himself.

A detection method in [26] uses blind signature cryptography with a publicly verifiable time-based payment transcript as double spending countermeasure. For the coin to be cashed by the client, the vendor must present a NIZK (non-interactive zero-knowledge) proof, which will bind the payment transcript to the target client and time. Another solution presented in [26] is a coin renewal protocol which provides a coin with three stages. Before reaching the dates, the coin can be cashed or renewed. If the coin reaches the first date, it can only be renewed. If it reaches the second date, the coin will be totally void.

3 System Design

3.1 Double Spending

To implement a double spending attack, the attacker first creates two transactions. The first transaction TV, transaction to vendor, lists the vendor as the recipient of the payment, and the second transaction TA, transaction to attacker, lists the attacker as the recipient of the payment. The attacker's goal is to have the vendor accept TV long enough to deliver the goods or services and have the rest of the network accept TA so that the attacker keeps the money.

The attacker sends out both transactions. TV is transmitted directly to the vendor, while TA is broadcasted to the rest of the network. In order for a double spending attack to be successful, 1) The attacker must know the vendor's IP address so it can connect to the vendor directly and send TV to the vendor; 2) The vendor must receive TV before TA arrives [20] to ensure that TA will be automatically dropped when the vendor eventually receives it; 3) TA must be confirmed in the block chain first or else TV will actually be confirmed and that block will become the accepted block in the network; 4) Given an equal propagation of both transactions, there is a 50 percent chance for either transaction to be confirmed. More nodes are required to work on TA than on TV to increase TA's likelihood of being accepted into the block chain, and it requires that the vendor only sees TV. Because the neighbors of the vendor will likely get TV first (directly from the vendor) and thus drop TA rather than propagate it to the vendor. This kind of double spending can succeed in fast-paying transactions in which the vendor does not wait for confirmation. Figure 1 shows 0-confirmation double spending.

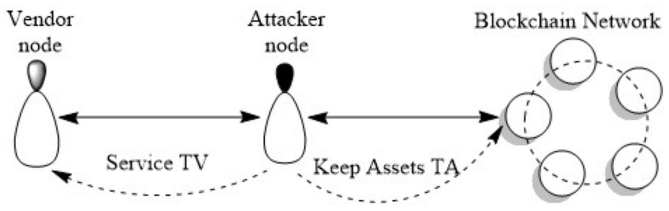

Figure 1: 0-confirmation Double Spending

Another form of double spending attack is block withholding attack [27], [28] in which the attacker pools resources to create a block BV, which contains TV. The attacker blocks all other connections to the vendor and prevents the vendor from ever receiving all other blocks confirming TA while sending BV the moment it is calculated. BV represents the block containing TV. The attacker essentially creates a fork in the block chain containing BV that will eventually be disregarded since no other mining pools work to extend this side of the fork [19]. This method of double spending can succeed in slow-pay transactions in which the vendor awaits confirmation. Figure 2 shows the N-confirmation double spending [16].

3.2 Design Assumptions

1. Our design is based on energy trading in smart grid and the payment methods can be tokens, money orders, checks, or any other payments that can be defined as unique and can be reusable.

2. Our double spending countermeasure is for slow-payment situations, such as paying electricity bill or buying renewable energy by consumers. We assume the attacker will try to use the same payment in at most two

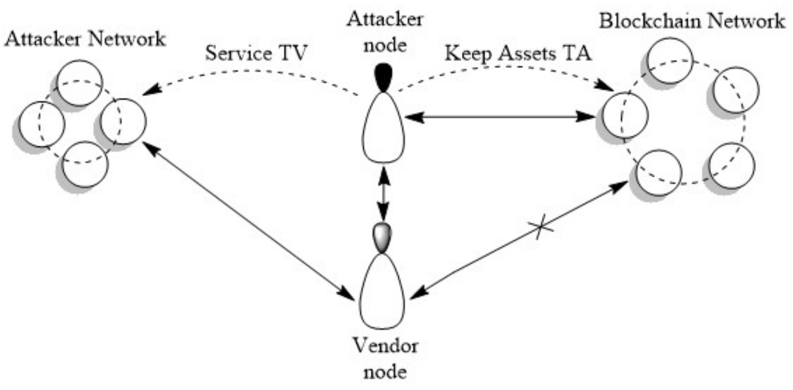

Figure 2: N-confirmation Double Spending

consecutive blocks, one is a previously validated block and the other is currently collecting transactions. If there are two conflicting transactions in one validated block, the transaction with the later timestamp will decide the block's final status.

3. Our design is based on our previous paper [29] and adds another reputation score, detection, to the reputation formula. The score of detection DT is cumulative and DT is calculated the same way as the voter's score.

$$DT_i = \pm Point \tag{1}$$

$$DT = \sum_{i=1}^{N} DT_i \tag{2}$$

The total reputation score is calculated with Offense contains all other attacking behaviors except double spending.

$$\begin{aligned} Reputation = {} & Resource + Defense + Availability \\ & + Offense + Service + Function \\ & + Detection + DoubleSpend \end{aligned} \tag{3}$$

4. When a new transaction arrives, every node checks if the transaction timestamp is later than current timestamp [16], if it is, this transaction is illegal and will be dropped. Otherwise, the transaction will be saved in the node's local memory pool.

5. We also use a similar consensus algorithm as in [29] by adding detection steps to the consensus in [29].

6. All the fields in the business transactions, except these two fields: the timestamp and the payment's receiver [29], are compared to decide if two transactions are conflicting or not.

7. The experimental environment is the same as in [29]

3.3 Double Spending Attack Models

1. Case 1: Suppose the attack only sends a vendor TV, then only the vendor's local block BV contains TV. When the vendor receives the block BL, which represents the block proposed by the leader, and finds out:

   $$BV \neq BL$$

   The vendor will fail BL. If BL is passed, the vendor will drop its local block BV and take BL. The attacker will not get the goods or service.

2. Case 2: TV is in a validated block. TA is added to the current block and waits for validation. The detector will discover TA is a conflicting transaction when comparing it with the transactions in the previously validated block. TA will be replaced with the transactions created by the detector (TDs). If BL is validated, the attacker will be penalized with the same amount payment it made in TA.

3. Case 3: If both TV and TA are put into one block, the transaction with the later timestamp will be discovered by the detector and replaced by TDs, which is created by the detector. The detector is rewarded with the same amount of payment in TA. If BL is validated, the attacker either gets the service or keeps its money. It also gets a penalty at the same time.

4. Case 4: The victim is the block leader, which is a special case for Case 1. BL is dropped, and the attacker failed its purpose. The detector is not aware of the attack and there will be no penalty to the attacker.

5. Case 5: The detector is the victim, another special case for Case 1. Any double spending will be discovered.

3.4 Double Spending Detection Procedure

Figure 3 shows the double spending detection flow chart.

1. At the beginning of each time interval, each node selects nodes with at least 90% of the highest total reputation score among all nodes as potential detector pool.

2. Select the node with the highest detection score from the potential detector pool as the detector.

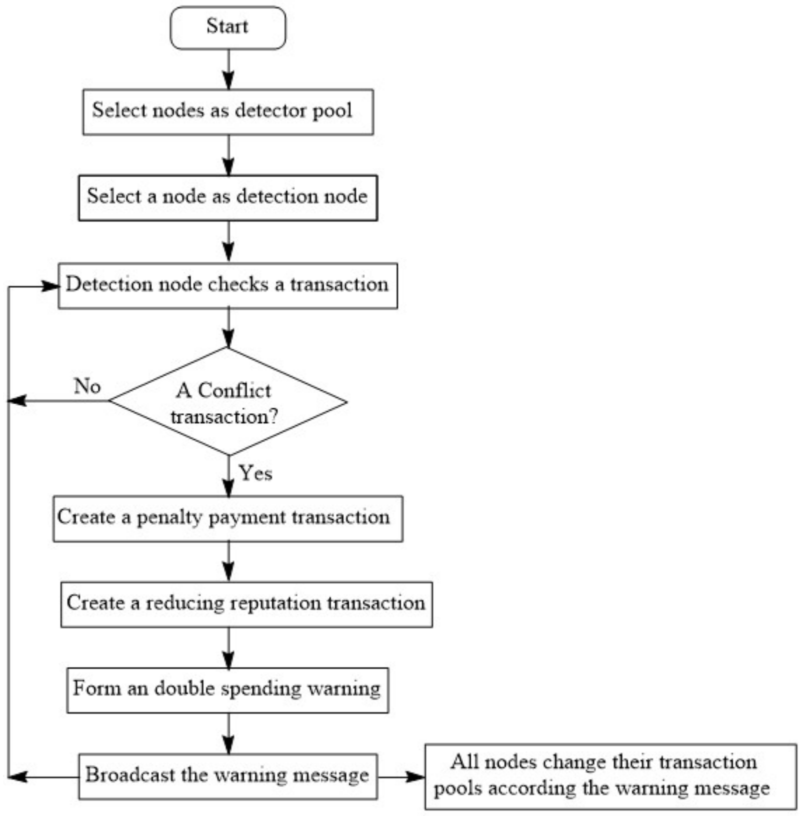
Start
Select nodes as detector pool
Select a node as detection node
Detection node checks a transaction
A Conflict transaction?
No
Yes
Create a penalty payment transaction
Create a reducing reputation transaction
Form an double spending warning
Broadcast the warning message
All nodes change their transaction pools according the warning message

Figure 3: Double-Spend Detection Flow Chart

3. The detector continuously checks and broadcasts conflicting transactions against the previous validated block and the transactions in current time interval. If an offending transaction is discovered, as a reward to the detector and a penalty to the attacker, the detector will create two transactions, one transaction will pay the same amount as in the conflicting transaction to the detector, the other one will be an offending reputation transaction to the attacker. The offending transaction and the two penalty transactions will form a warning message and be broadcasted to all nodes in the network.

4. Upon receiving the detector's warning message, all nodes drop the offending transaction and add the detector's reward transaction and the attacker's offending reputation transaction in their memory pools.

Upon finishing the consensus, all nodes update the offending node's scores accordingly, and update the detector's reputation scores the same way as updating a node's voting scores.

Table 1: Detection and Consensus Performance

Testing	Number of Transactions	Time of Consensus (s)	Time of Detection (s)	Status of Block	Double Spending Detected ?	Size of Block
1	7	5	5	Passed	Yes	4KB
2	53	5	5	Passed	Yes	32KB
3	103	5	5	Passed	Yes	58KB
4	203	5	5	Passed	Yes	115KB
5	403	6	5	Passed	Yes	227KB
6	803	8	5	Passed	Yes	452KB
7	1203	12	5	Passed	Yes	677KB
8	1603	18	5	Passed	Yes	902KB
9	2003	19	5	Passed	Yes	1.1MB

4 Experimental Results and Analysis

4.1 Experimental Transaction Creation

Figure 4 shows how the experimental transactions were created. When it is time to simulate, a loop number was set, which controls the number of transactions to be created. In each loop, one business transaction and one reputation transaction were created.

Figure 5 shows the case testing flow chart. In each testing case, the testing transactions were either sent to a specific node or were broadcasted to the entire network.

4.2 Detection and Consensus Performance

Table 1 and Figure 6 display the experimental data and graph, respectively. The normal detection time was five seconds for one double spending detection, which was not affected by the number of transactions in the block. The time was just necessary to run the detection program, which performed much better than all the three cases in [23]: of All ENHOBS, 1% Skinny, and 2% Skinny. The consensus time did not change until the number of transactions reached 400. The pattern and values of the consensus are like the results we obtained previously [29]. This is expected because the consensus was conducted in a similar way.

Our approach used much less CPU time than [23]. Our detection node used 0.1% of CPU, while with every node acting as observers in [23], the CPU utilization jumped from 31% to 50.6%, with the maximum CPU utilization reaching as high as 96%. This is understandable because one detection node will use much less resources than many nodes as detection nodes at the same time.

We tested all five double spending cases. The detection rate is 100% for C2, C3 and C4, and 0 for C1 and C5, as shown in Figure 7. We tested the detection rates with blockchain standard maximum block size 1.1 MB, which is equivalent to 2000 transactions. Worth noticing is that the detection rate is 0 when the double spending victim is the vendor or the block leader. This is because the

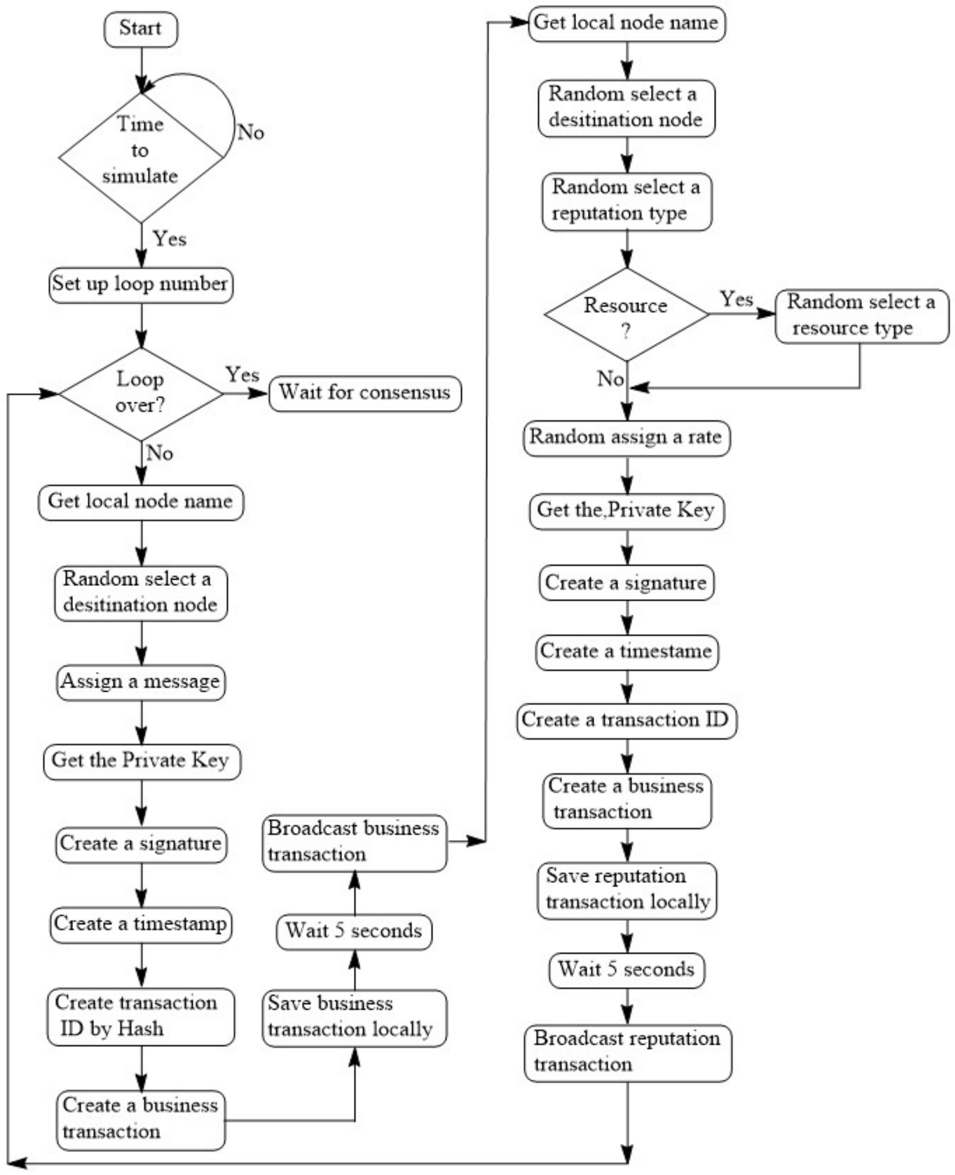
Start
Time to simulate
No
Yes
Set up loop number
Loop over?
Yes
Wait for consensus
No
Get local node name
Random select a desitination node
Assign a message
Get the Private Key
Create a signature
Create a timestamp
Create transaction ID by Hash
Create a business transaction
Save business transaction locally
Wait 5 seconds
Broadcast business transaction
Get local node name
Random select a desitination node
Random select a reputation type
Resource ?
Yes
Random select a resource type
No
Random assign a rate
Get the,Private Key
Create a signature
Create a timestame
Create a transaction ID
Create a business transaction
Save reputation transaction locally
Wait 5 seconds
Broadcast reputation transaction

Figure 4: Transaction Creation Flow Chart

conflict transaction was only sent to the vendor or leader and the detector did not receive TA. In both situations, the local block/blocks were not the same as the proposed block. In these two situations, more system resources were wasted when the leader was the victim than when non-leader node was attacked. This is because non-leader node only needs to drop its local block while the leader's

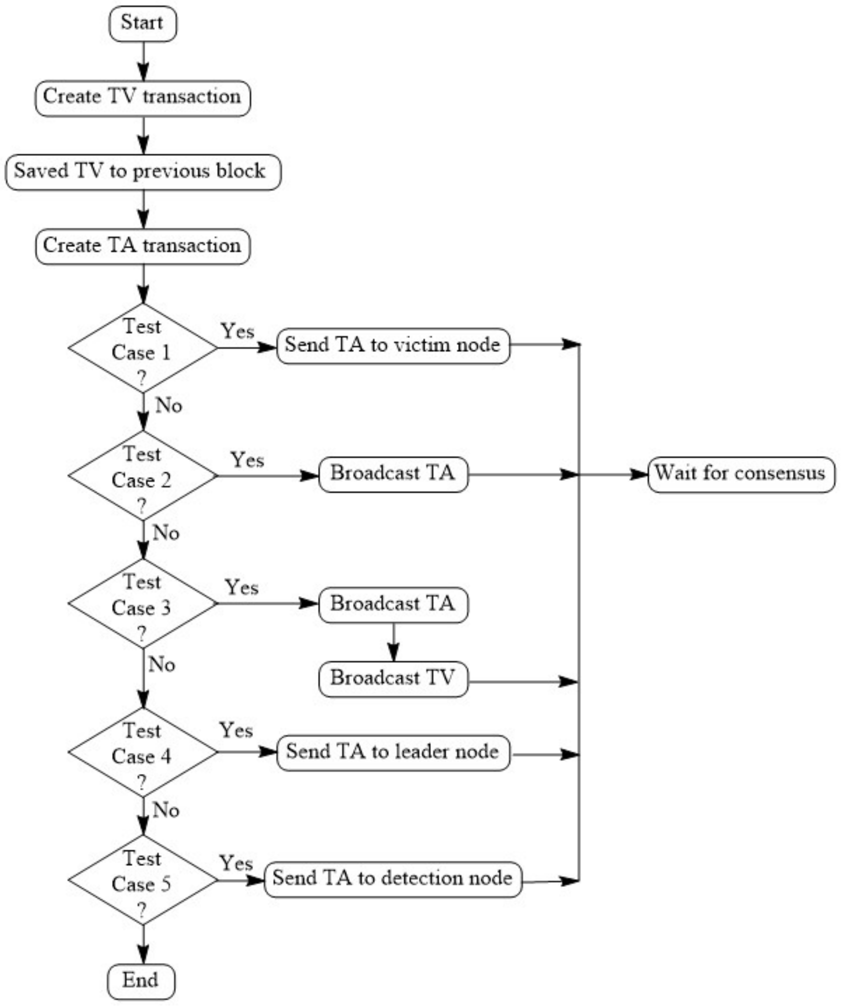

Figure 5: Double-Spend Case Test Flow Chart

block was dropped after all the nodes in the network had verified it.

4.3 Detection Complexity

The complexity of the double spending detection is O(N), where N is the number of transactions in a block. The conflicting transaction will be searched throughout the previously validated block and current block, so the detection time should be 2O(N). The complexity of consensus algorithm is $O(N^2)$ because the transactions in two block are compared.

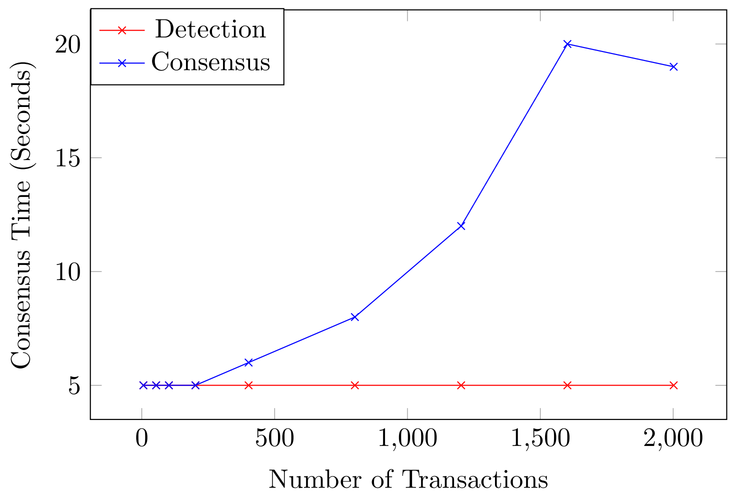

Figure 6: Consensus and Detection Performance

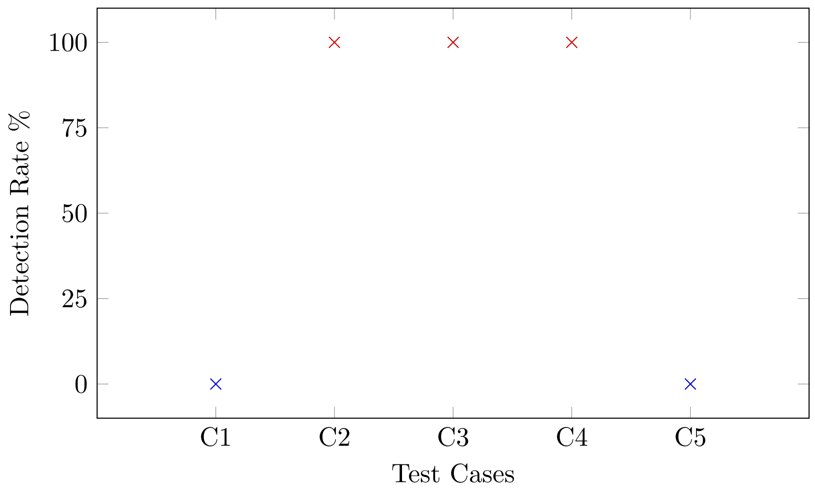

Figure 7: Double-Spend Detection Effectiveness

4.4 Security Analysis

1. TV was only sent to the vendor, whereas other nodes received TA. The block BA, which contained TA, was validated, and the vendor dropped its local block BV, which contained TV. The attack failed to get service and did not get penalized either because the attack was not uncovered.

2. When TV was included in the previous block, and TA was in current block, the detector found TA was a conflicting transaction and sent a rewarding transaction to itself and a reducing reputation transaction to the attacker. The block, which did not contain TA, was validated by all nodes. The attacker did not recover its payment.

3. When TV was in current block already and TA was broadcasted again, the detector found TA and sent a rewarding transaction to itself and a reducing reputation transaction to the attacker. The block containing TV was validated. The attacker only received service.

4. When the leader was attacked, the detection failed because the offending transaction TA was only sent to the leader. The proposed block BL by the leader failed because other nodes didn't have the same transactions as the leader. No damage was done to any nodes but the system resources were wasted. The attacker only received service and failed to get its money back.

5. When the detector was attacked, the double spending detection rate was 100%, this was because the detector had both TA and TV. Beside failing to get its money back, the attacker got two penalty transactions, payment to the detector and lost its reputation scores.

5 Conclusion

In this paper, we proposed a double spending countermeasure, which can effectively detect double spending in two consecutive blocks. Our design puts detection results into the consensus mechanism, handles the offending transaction, and has a mechanism to prevent the perpetrator to double spend again. Comparing to other countermeasures, such as time period monitoring and inserting observers, our method constantly monitors transactions, uses less computing resources, and reduces network traffic and the management overhead. Another advantage of our design is our detection node is not fixated and it does not have a specific communication pattern, which will less likely attract DDoS attack. However, there are some limitations for the current research. For example, because it only checks the conflicting transactions in two consecutive blocks, it will not be able to detect double spending if a transaction confirmation is larger than 1. Another limitation is the single detector might behave maliciously, or it might not be able to process all transactions if the number of transactions is huge. Future research will: 1) handle the situation when the block leader is attacked; 2) be checking all kinds of transactions, such as sybil attack, self-mining attack, business, and service transactions; 3) expand the detection to the entire blockchain to overcome N-confirmations double spending; 4) use a set of detector to check the security attack transactions. Of the four future directions, 2 and 3 are parts of the reasons we chose to have a single node to perform the detection duties.

References

[1] Tatsuya Sato, Yosuke Himura. Smart-contract based system operations for permissioned blockchain. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–6, 2018.

[2] Kervins Nicolas, Yi Wang, George C. Giakos, Bingyang Wei, Hongda Shen. Blockchain system defensive overview for double-spend and selfish mining attacks: A systematic approach. *IEEE Access*, 9:3838–3857, 2021.

[3] Zhaoyang Dong, Fengji Luo, Gaoqi Liang. Blockchain: a secure, decentralized, trusted cyber infrastructure solution for future energy systems. *Journal of Modern Power Systems and Clean Energy*, 6:958—-967, Jul. 2018.

[4] Yunsen Wang and Alexander Kogan. Designing confidentiality-preserving blockchain-based transaction processing systems. *International Journal of Accounting Information Systems*, 30:1–18, September 2018.

[5] Dr Mahdi H. Miraz, David Donald David. Application of blockchain in booking and registration systems of securities exchanges. *in Proc. Int. Conf. Comput., Electron. Commun. Eng. (iCCECE)*, page 35–40, August 2018.

[6] Feng Gao, Liehuang Zhu, Meng Shen, Kashif Sharif, Zhiguo Wan, Kui Ren. A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks. *IEEE Network*, 32(6):184–192, 2018.

[7] Lin Zhong, Qianhong Wu, Jan Xie, Jin Li, Bo Qin. A secure versatile light payment system based on blockchain. *Future Generation Computer Systems*, 93:327–337, 2019.

[8] Lei Xu, Lin Chen, Zhimin Gao, Larry Carranco, Xinxin Fan, Nolan Shah, Nour Diallo, Weidong Shi. Supporting blockchain-based cryptocurrency mobile payment with smart devices. *IEEE Consumer Electronics Magazine*, 9(2):26–33, 2020.

[9] Ahmed S. Musleh, Gang Yao, S. M. Muyeen. Blockchain applications in smart grid–review and frameworks. *IEEE Access*, 7:86746–86757, 2019.

[10] Keke Gai, Yulu Wu, Liehuang Zhu, Meikang Qiu, Meng Shen. Privacy-preserving energy trading using consortium blockchain in smart grid. *IEEE Transactions on Industrial Informatics*, 15(6):3548–3558, 2019.

[11] Mohamed Amine Ferrag, Leandros Maglaras. Deepcoin: A novel deep learning and blockchain-based energy exchange framework for smart grids. *IEEE Transactions on Engineering Management*, 67(4):1285–1297, 2020.

[12] Cong T. Nguyen, Dinh Thai Hoang, Diep N. Nguyen, Dusit Niyato, Huynh Tuong Nguyen, Eryk Dutkiewicz. Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities. *IEEE Access*, 7:85727–85745, 2019.

[13] N Anita., M Vijayalakshmi. Blockchain security attack: A brief survey. pages 1–6, 2019.

[14] Mubashar Lqbal, Raimundas Matulevičiusi. Blockchain-based application security risks: A systematic literature review. *In Advanced Information Systems Engineering Workshops*, page 176–188., 2019.

[15] Kervins Nicolas, Yi Wang. A novel double spending attack countermeasure in blockchain. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0383–0388, 2019.

[16] Mubashar Iqbal, Raimundas Matulevičius. Exploring sybil and double-spending risks in blockchain systems. *IEEE Access*, 9:76153–76177, 2021.

[17] Aziz Mohaisen, oongheon Kim. The sybil attacks and defenses: A survey. *Smart Computing Review*, 3:1–10., 2013.

[18] Ghassan O. Karame, Elli Androulaki, Srdjan Capkun. Double-spending attacks on fast payments in bitcoin. page 906–917, 2012.

[19] G. O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, and S. Capkun. Misbehavior in bitcoin: A study of double-spending and accountability. *ACM Transactions on Information and System Security (TISSEC)*, 18, June 2015.

[20] Tobias Bamert, Christian Decker, Lennart Elsen, Roger Wattenhofer, Samuel Welten. Have a snack, pay with bitcoins. In *IEEE P2P 2013 Proceedings*, pages 1–5, 2013.

[21] Rosenfeld, Meni. Analysis of hashrate-based double spending. *arXiv e-prints*, page arXiv:1402.2009, Feberary 2014.

[22] Matthias Grundmann, Till Neudecker, Hannes Hartenstein. Exploiting transaction accumulation and double spends for topology inference in bitcoin. In *Financial Cryptography and Data Security*, pages 113–126, 2019.

[23] John P. Podolanko, Jiang Ming. Countering double-spend attacks on bitcoin fast-pay transactions. 2017.

[24] Micha Ober, Stefan Katzenbeisser, Kay Hamacher. Structure and anonymity of the bitcoin transaction graph. *Future Internet*, 5:237–250, 2013.

[25] Cristina Perez-Sola, Sergi Delgado-Segura, Guillermo Navarro-Arribas, Jordi Herrera-Joancomarti. Double-spending prevention for bitcoin zero-confirmation transactions, 2017.

[26] Ivan Osipkov, Eugene Y. Vasserman, Nicholas Hopper, Yongdae Kim. Combating double-spending using cooperative p2p systems. In *27th International Conference on Distributed Computing Systems (ICDCS '07)*, pages 41–41, 2007.

[27] Arthur Gervais, Hubert Ritzdorf, Ghassan O. Karame, Srdjan Capkun. Tampering with the delivery of blocks and transactions in bitcoin. *in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS'15)*, page 692–705, October 2015.

[28] Samiran Bag, Sushmita Ruj, Kouichi Sakurai. Bitcoin block withholding attack: Analysis and mitigation. *IEEE Transactions on Information Forensics and Security*, 12(8):1967–1978, 2017.

[29] Lanqin Sang and Henry Hexmoor. Reputation-based consensus for blockchain technology in smart grid.