**16010121017 - Viraj Bhansali**

**16010121033 - Nupur Chaudhari**

**Bct IA 1**

**A\_Review\_on\_Double\_Spending\_Problem\_in\_Blockchain**

https://ieeexplore.ieee.org/document/10183579

**Abstract**

Double spending is a major security problem in blockchain, where the same digital currency is used more than once, which can damage the trust and security of blockchain systems. This report looks at how double spending happens, examines how consensus methods like Proof of Work (PoW) and Practical Byzantine Fault Tolerance (PBFT) try to solve the issue, and compare them.

**Existing literature on the security issue(s) identified in the approved paper.**

Double spending occurs when a user spends the same cryptocurrency more than once, posing a serious risk to the trustworthiness of blockchain systems. Existing research points to several factors contributing to this issue, including network latency, malicious actors, and the design of consensus mechanisms like Proof of Work (PoW), Proof of Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT).

While PoW and PoS are widely used, they are not foolproof. PBFT, on the other hand, is designed to address the problem in a different way. It works by having multiple nodes (or computers) agree on the validity of transactions, requiring a majority vote to confirm any transaction. This helps prevent double spending by ensuring that even if some nodes are compromised, the network can still function securely. However, PBFT is more communication-heavy, making it less efficient for larger networks. Emerging technologies such as zero-knowledge proofs and multi-party computation are also being explored to further strengthen the security of blockchain systems​

**Impact of Double Spending in Blockchain Applications**

* **Financial Losses**: Double spending can result in significant financial losses for users and businesses that rely on blockchain-based transactions.
* **Erosion of Trust**: If double-spending incidents increase, users may lose trust in the security and reliability of blockchain systems, particularly in cryptocurrencies.
* **Reversed Transactions**: Malicious actors can exploit vulnerabilities in consensus mechanisms to reverse transactions, leading to disputes and loss of funds.
* **Disruption of Decentralized Applications (dApps)**: Blockchain applications that rely on secure and transparent transactions can be severely impacted, causing disruptions in their services.
* **Economic Instability**: Frequent double-spending attacks can undermine the value of digital currencies, creating instability in blockchain-based financial systems.
* **Damage to Blockchain's Reputation**: Continuous double-spending issues may harm blockchain’s reputation as a secure and transparent technology, slowing down its adoption across industries

**Potential solutions to mitigate/overcome the security challenges.**

1. **Longer Confirmation Times (PoW)**: Increasing the number of confirmations required before considering a transaction final can reduce the risk of double spending by making it harder for attackers to reverse transactions.
2. **Transaction Locking**: Implementing time-lock mechanisms that temporarily "freeze" transactions before they are confirmed can minimize the chances of double spending during network delays.
3. **Quorum Requirements (PBFT)**: In PBFT, requiring a supermajority (often two-thirds) of nodes to validate transactions makes it difficult for malicious actors to gain control and execute double-spending attacks.
4. **State Machine Replication (PBFT)**: Ensures that all nodes maintain a consistent state, reducing the risk of conflicting transactions, thereby minimizing the chances of double spending.
5. **Zero-Knowledge Proofs**: This technology allows verification of transactions without revealing sensitive data, providing enhanced security and preventing unauthorized actions like double spending.
6. **Multi-Party Computation (MPC)**: Allows multiple parties to jointly verify a transaction without compromising privacy, adding another layer of defense against double spending attempts.
7. **Network Observers**: These can monitor for unusual activity, such as repeated transactions from the same user, and block suspicious transactions to prevent double spending.
8. **Peer Warning Systems**: Alerts nodes of any suspicious activity or fraudulent transactions, enabling them to cut off connections with the malicious nodes and reroute transactions securely

We will explore the PBFT and PoW solutions in the assignment.

**PoW solution**

1. **Consensus Mechanism**: In PoW, miners compete to solve complex mathematical problems, and the first to solve it gets to add a new block to the blockchain. This process helps to secure the network and ensures that all nodes agree on the current state of the ledger.
2. **Longest Chain Rule**: The blockchain is maintained as a chain of blocks, with the longest chain being considered the valid one. If a miner attempts to double spend, they would need to create an alternate chain that is longer than the one that includes the original transaction, which requires significant computational power and resources.
3. **Chain Structure**: A blockchain consists of blocks linked together. Each block contains a set of transactions and a reference (hash) to the previous block.
4. **Consensus on the Longest Chain**: When nodes in the network receive conflicting versions of the blockchain (e.g., due to double spending), they choose the chain that has the most work done, i.e., the longest chain in terms of blocks or the heaviest chain in terms of cumulative computational effort.
5. **Network Propagation**: Once a transaction is broadcasted, it spreads through the network. Miners and nodes validate the transaction against their current ledger. If a double spending attempt occurs, only one of the conflicting transactions will be included in the blockchain, depending on which block is mined first.
6. **Transaction Confirmations**: Users typically wait for multiple confirmations before considering a transaction final. This means that the more blocks that are added after a transaction, the harder it becomes to alter that transaction, as it would require re-mining not just the block containing the transaction but all subsequent blocks.
7. **Incentives for Honesty**: Miners are incentivized to act honestly because attempting to double spend is costly. They risk losing their investment in hardware and energy if they are detected and penalized by the network.

### **Example**

Imagine the following scenario:

* **Transaction A**: Alice sends 1 Bitcoin (BTC) to Bob.
* **Transaction B**: Alice tries to send the same 1 BTC to Charlie (double spending).

#### **Step 1: Two Blocks Created**

1. **Block 1**: Contains Transaction A (Alice to Bob) and is mined first.
2. **Block 2**: Contains Transaction B (Alice to Charlie) and is mined shortly after Block 1, creating a fork in the blockchain.

Block 1 (A to Bob)

|

Block 2 (B to Charlie)

#### **Step 2: Propagation**

* Nodes in the network receive both blocks, leading to two versions of the blockchain:
  + Version 1: Block 1 (A to Bob)
  + Version 2: Block 2 (B to Charlie)

#### **Step 3: Mining Continues**

Now, miners start mining the next block:

* **Miner 1** mines on top of Block 1, creating Block 3 (continuing the chain with Bob's transaction).
* **Miner 2** mines on top of Block 2, creating Block 4 (continuing the chain with Charlie's transaction).

Version 1: Block 1 (A to Bob) -> Block 3

|

Version 2: Block 2 (B to Charlie) -> Block 4

#### **Step 4: Longest Chain Rule**

Suppose Block 3 is mined first and broadcasted to the network, causing nodes to see:

* **Chain A**: Block 1 -> Block 3 (2 blocks total)
* **Chain B**: Block 2 -> Block 4 (2 blocks total)

If more miners continue to mine on Chain A and it eventually gets more blocks added (say Block 5), Chain A would become the longest chain:

Block 1 -> Block 3 -> Block 5 (A to Bob)

|

Block 2 -> Block 4 (B to Charlie)

#### **Step 5: Final Consensus**

Now, most nodes will adopt Chain A as the valid chain because it has the most blocks (or the most cumulative work), confirming Transaction A (to Bob) and rejecting Transaction B (to Charlie).

### **PBFT Solution for Preventing Double Spending**

1. **Pre-Prepare Phase (Leader Proposes a Block)**:
   * A primary (leader) node proposes a block of transactions, including any new transaction (e.g., Alice sends 1 BTC to Bob).
   * The block is broadcast to all other nodes (replicas) for validation.
2. **Prepare Phase (Nodes Validate the Proposal)**:
   * All replicas receive the proposed block and check if it contains valid transactions, including whether Alice actually has 1 BTC to spend.
   * The replicas ensure that Alice’s transaction hasn’t been spent already.
   * Each node sends a "prepare" message to all other nodes indicating that it has received and validated the proposal.
3. **Commit Phase (Nodes Agree on the Block)**:
   * Once a node receives a sufficient number (2/3 majority) of "prepare" messages, it moves to the commit phase.
   * Nodes broadcast "commit" messages, essentially saying, "I agree to add this block to the ledger."
   * Once a node receives a sufficient number of "commit" messages, it finalizes the block and adds it to its local blockchain.
4. **Consensus and Finality**:
   * In PBFT, consensus requires agreement from 2/3 of the nodes.
   * Once a block is added, it is considered final and irreversible. Double spending is impossible because:
     + All valid nodes must agree on the same version of the block.
     + Any attempt to submit two conflicting transactions (e.g., Alice sending the same 1 BTC to both Bob and Charlie) will be detected by the nodes during the validation phase. They will reject one of the transactions since Alice cannot spend the same 1 BTC twice.

### **Example of PBFT Preventing Double Spending**

* **Transaction A**: Alice sends 1 BTC to Bob.
* **Transaction B**: Alice tries to send the same 1 BTC to Charlie (double spending).

1. **Leader Proposes Transaction A**: The leader node proposes the block with Transaction A (Alice to Bob) and broadcasts it to the network.
2. **Prepare Phase**: All nodes validate Transaction A, checking Alice’s account balance, and ensuring she has enough funds to send 1 BTC.
3. **Commit Phase**: Nodes reach a consensus and finalize the block with Transaction A.
4. **Double Spending Detection**: If Alice later tries to submit Transaction B (sending the same BTC to Charlie), the nodes will reject it because they’ve already confirmed Transaction A. Alice’s 1 BTC is no longer available.

**Comparison of PoW and PBFT**

| Aspect | Proof of Work (PoW) | Practical Byzantine Fault Tolerance (PBFT) |
| --- | --- | --- |

| Consensus Mechanism | Solves complex mathematical puzzles to validate transactions | Nodes communicate and agree through voting; requires a supermajority (two-thirds) to validate transactions |
| --- | --- | --- |

| Energy Consumption | High energy consumption due to mining | Low energy consumption as no mining is involved |
| --- | --- | --- |

| Security Model | Security relies on computational power; vulnerable to 51% attacks | Resilient to malicious nodes as long as less than one-third of nodes are faulty or compromised |
| --- | --- | --- |

| Scalability | Scales well in large, open networks (e.g., Bitcoin) | Works best in small to medium-sized networks; scalability decreases with more nodes due to high communication overhead |
| --- | --- | --- |

| Transaction Speed | Slower transaction speed due to mining and multiple confirmations | Faster as transactions are validated by voting rather than mining |
| --- | --- | --- |

| Finality | Probabilistic finality; transactions can be reversed if an attacker controls 51% of the network | Immediate finality; once a block is validated, it cannot be reversed |
| --- | --- | --- |

| Vulnerability | Vulnerable to 51% attacks (if one entity controls more than half the mining power) | Vulnerable if more than one-third of the nodes are malicious or faulty |
| --- | --- | --- |

| Use Cases | Widely used in public blockchains like Bitcoin and Ethereum | Typically used in permissioned or consortium blockchains (e.g., Hyperledger) |
| --- | --- | --- |

| Communication Overhead | Low; nodes do not need to communicate with each other beyond mining | High; requires constant communication between all nodes to reach consensus |
| --- | --- | --- |

**Conclusion.**

Double spending remains a crucial challenge in blockchain security, particularly for digital currencies and applications that require immutable and trustless transactions. While PoW has proven resilient, it remains vulnerable to 51% attacks, while PBFT offers a more secure but complex alternative. Addressing the vulnerabilities in both models through enhanced security measures is essential for maintaining the trust and stability of blockchain systems.