Strengthening Fawkescoin against Double Spending Attack using Merkle Tree

Widya Wirachantika
Telkom University
wirachantika@student.telkomuniversity.ac.id

Ari Moesriami Barmawi
Telkom University
mbarmawi@melsa.net.id

Bambang Ari Wahyudi
Telkom University
bambangari@telkomuniversity.ac.id

ABSTRACT
Cryptocurrency is a digital currency with cryptographic security so that it is not easily to be faked. Recently cryptocurrency is widely used for transactions. Therefore, preserving its integrity and security is important. The technology underlying the digital currency is the Blockchain, as applied to Fawkescoin. But for securing the transaction, fawkescoin has a disadvantage when the fork occurs because it can provide opportunity to conduct double spending attack. To overcome this problem, Merkle tree proposed by applying DSA digital signatures. The application of DSA on Merkle tree is used to verify data without knowing the contents of the data. Based on the experiment results and analysis, the security of the proposed method can prevent fawkescoin against double spending attack on transactions when forking occurs than the previous method.

CCS Concepts
• Security and privacy~Hash functions and message authentication

Keywords
Cryptocurrency; Blockchain; Double Spending Attack; Merkle Tree.

1. INTRODUCTION
Cryptocurrency is a digital currency with cryptographic security so that it is not easily to be faked. Since cryptocurrency is frequently used for bussiness transaction then it important to preserve the integrity and confidentiality of transaction. The technology underlying the digital currency is the Blockchain. Blockchain is a distributed database consists of transactions data history that are executed and shared between participants. The blockchain structure consists of transactions and blocks, where the transaction is exchanged between participants and a block is a collection of transaction data and other related information. The objective of blockchain is to create a node that are connected to each other based on their spending conditions to prevent double spending [1]. One of the digital currencies that implements the blockchain is Fawkescoin.

ICCSP 2019, January 19–21, 2019, Kuala Lumpur, Malaysia

ACM ISBN 978-1-4503-6618-2/19/01…$15.00

DOI: https://doi.org/10.1145/3309074.3309105

Fawkescoin is a simple cryptocurrency constructed using hash-based Guy Fawkes signature [2]. Guy fawkes signature is one simple digital signature algorithm that used to authenticate transaction based on one way hash function. However it has security problem when forking is applied on fawkescoin that is double spending attack [3]. A fork is a branching on the blockchain with a different protocol application so that if after forking process other block is found then the chain would be longer. Furthermore, the branch will become a part of the main blockchain [4]. When this condition occurred, all valid transactions from the shorter chain (the orphaned block) are added to the pool of queued transaction [1]. If this occurs, the attacker can spend the transaction more than once using the same coin in each transaction history that appears on the overwritten block, namely double spending attack [5]. Double sppending attack is occurred because the user revealing the transaction in the block as finalize message [3].

For overcoming the problem, a digital signature algorithm (DSA) (as shown in Section 3.1) and Merkle trees (as shown in Section 3.2) we proposed. Digital signature algorithm (DSA) is a public key cryptographic scheme to verify the transactions [6]. DSA is used to verify the transactions on each block without knowing the content. DSA is implemented using private key. Each signature will be the input node of the Merkle tree. The Merkle tree is a binary tree that implements hash functions as the integrity of each data entered in the block [1]. As the impact of the proposed method, the attacker cannot arbitrarily change and double-spend the transaction on the overwritten block.

2. FAWKESCOIN
Fawkescoin consisted of three sub-process; mining, transfer, finalize. For mining the coin with a value v, we need to generate a secret codeword X, known only to the owner and hash it as the address of the block owner H(X). To transfer the value from address H(X) to address H(Y), the owner needs to construct the transfer message T_i : H(X ,H(Y)) which means that the owner of X is sending to address H(Y) and publish it into the blockchain as a proof that the message was transferred by the legal owner. Furthermore, the owner must wait until the transfer message T_i is on the blockchain, then the transfer message T_i was revealed as the finalize message F_i : (X ,H(Y)). This process is necessary for preventing a temporary fork. This message allows user to verify the transfer message T_i and know the current X value, so that the address H(X) cannot be used as the owner address in the next block. The overview of Fawkescoin as shown in Figure 1.

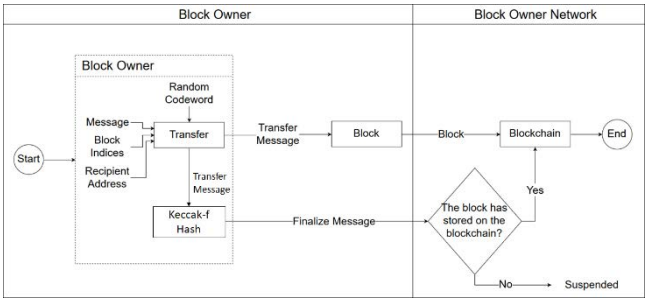

Figure 1. Fawkescoin

For constructing the block of Fawkescoin protocol as shown in Figure 2 was proposed. In this section, we use the following notations and conventions. This protocol shows the construction of transaction blocks with the block index b_i , that contains transactions whose transfer messages T_i have been revealed. Each block has information that is stored on the blockchain Bc_i such as including previous hash codeword $prevHc_i$, the constructed time t_i, nonce $nonce_i$, current hash codeword Hc_i, and proof of work PoW_i. If the hash of previous codeword and proof of work is $false$, then the block B_i will be rejected by the network, else the block B_i will be added by the network to be the new block in the blockchain $Bc_i \leftarrow B_{i-1}, B_i$. Furthermore, block owner revealed the transfer message F_i to the blockchain Bc_i using finalize message F_i.

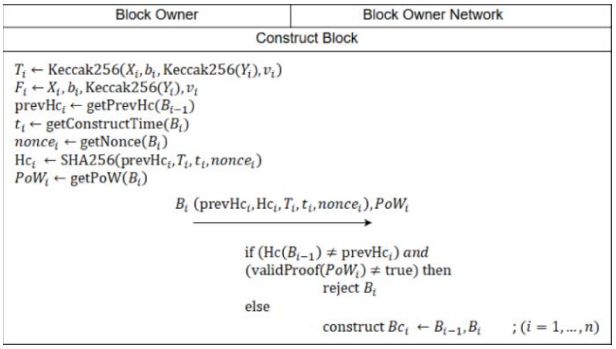

Figure 2. Constructing Fawkescoin Block

3. PROPOSED METHOD

The proposed method consisted of the two processes; digital signature algorithm (DSA) [6][7] and Merkle tree [8]. DSA consisted of three sub-processes; key creation, signing, and verification. This signature will be used as input node to the Merkle tree and it is hashed with the partner node to get the Merkle root value. This root value is used to verify whether each transaction contained in the block was changed or not, and to check whether the block contained in the block chain is the actual block or not. The overview of our proposed method is shown in Figure 3.

3.1 Protocol Design

Following that structure, as shown in Figure 3, we present the protocol design for constructing the block from our proposed method. Different from the previous protocol, as shown in Figure 2, in our protocol for constructing a block is shown in Figure 4, the user constructed a digital signature d_i with a finalize message F_i as the input message. Send the digital signature d_i and a transfer message was sent by the user to the block owner. The block owner collect digital signatures and transfer messages from each user. Blocks B_i had information about the previous hash block $prevHb_i$, current block hash Hb_i, transfer message T_i, Merkle tree root M_{r_i}, block constructing time t_i, nonce $nonce_i$ and proof of work PoW_i. Before the block is added to the blockchain Bc_i, it will check whether the added block has the right previous block hash and valid proof of work. If the previous hash block and proof of work is $false$, then the block B_i will be rejected by the network, else the block B_i will be added by the network to be the new block in the blockchain $Bc_i \leftarrow B_{i-1}, B_i$.

3.2 Digital Signature Algorithm (DSA) for Signing and Verifying Finalize Message

This section describes about generating the DSA as shown in Figure 5. In the proposed method, DSA will be implemented to secure the finalize message F_i when revealing the transaction in the blockchain.

3.2.1 Key Generation

For signing the finalize message, the sender should generate public parameters and the key. For determining the public parameters and the key, the sender choose prime number p determined by L bits long, where L range should be 512-bit to 1024-bit and it is a multiple of 64, a prime number q determined by a 160-bit prime factor of $p-1$ and calculate g using Eq.1.

$$g = h^{(p-1)/q} \bmod p \tag{1}$$

where h is a primitive root of p. The primes of p, q and g was public parameters. The secret key x is was a random number less than q and calculate the public key y using Eq.2.

$$y = g^x \bmod p \tag{2}$$

3.2.2 Generating Signature

This process was used for generating the digital signature used to sign message was a finalize message from the transaction. For signing the finalize message F_i, the signer chose a random number k in the range $1 \leq k < q$ as the ephemeral key, where k should have multiplicative inverse and computes the two value r and s using Eq.3 and Eq.4 respectively.

$$r = (g^k \bmod p) \bmod q \tag{3}$$

and

$$s = \left(k^{-1}(\mathrm{H}(F_i) + xr)\right) \bmod q \tag{4}$$

The digital signature of the message, F_i, is (r, s) and it was published in the block. Furthermore, this signature was the input node of Merkle tree, as shown in Section 3.2.

3.2.3 Verification

The verification process was used to verify the signature (as shown in Section 3.1) sent by the sender. The verification process was carried out if there was a user who needed to verify the transaction in the block. The user verified the signature by computing the u_1 and u_2 using Eq.5 and Eq.6.

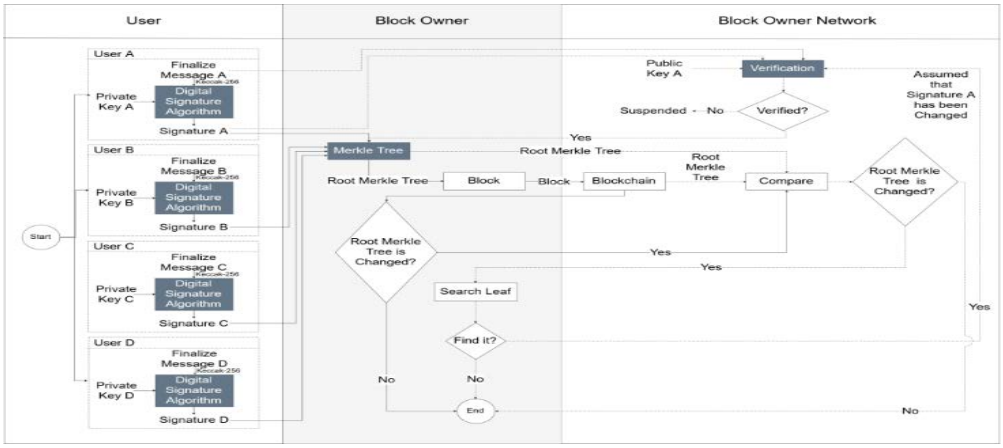

Figure 3. Fawkescoin with Digital Signature Algorithm and Merkle Tree

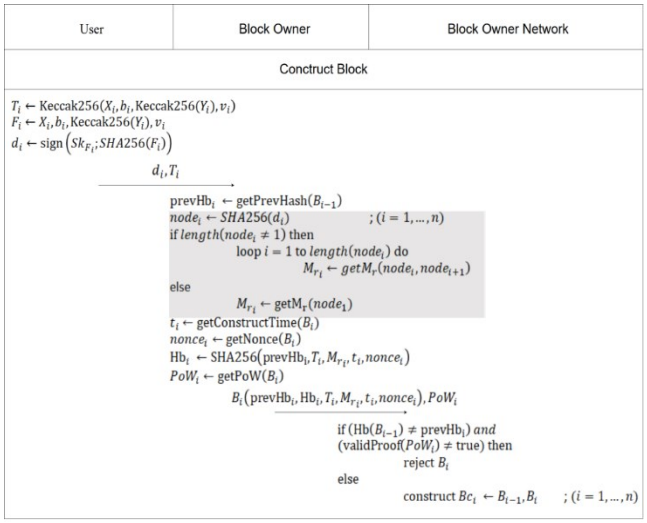

Figure 4. Block Constructor Protocol

User	Other User
Key Generation	
• She chooses two large random primes p and q, then computes an element $g = h^{\left(\frac{p-1}{q}\right)} \bmod p$, where h is a primitive root of p • Her secret key is x, where $1 \leq x < q$ • She computes the public key $y = g^x \bmod p$ • She publishes the public key y	
Generating Signature	
• She calculates the finalize message F_i • She chooses the ephemeral key k, where $1 \leq k < q$ • She computes signature $r = \left(g^k \bmod p\right) \bmod q$ $s = \left(k^{-1}(\text{SHA256}(F_i) + xr)\right) \bmod q$	
Verification	
	• He computes $u_1 = (\text{SHA256}(F_i) * s^{-1}) \bmod q$ $u_2 = (r * s^{-1}) \bmod q$ $r' = (g^{u_1} y^{u_2} \bmod p) \bmod q$ • He verifies that $r = r'$

Figure 5. Digital Signature Algorithm

$$u_1 = \left(\text{H}(F_i) * w\right) \bmod q \tag{5}$$

and

$$u_2 = (r * w) \bmod q \tag{6}$$

Furthermore, the sender calculate the verification parameter r' using Eq.(7)

$$r' = \left(\left(g^{u_1} * y^{u_2}\right) \bmod p\right) \bmod q \tag{7}$$

If $r' = r$, then the signature was valid and the sender was successfully verified. Otherwise, the verification process of signature would be suspended. This verification process is shown in Figure 5.

3.3 Merkle Tree for the Digital Signature Algorithm (DSA)

This section describes the Merkle tree constructing with height h as shown in Algorithm 1. Let the signature d_i represent the user's signature (as shown in Section 3.2) which was the input node of the Merkle tree. This method connect hash leaf n_i with the partner leaf n_{i+1} by applying $\text{H}(n_i, n_{i+1})$ using a one-way hash function, SHA256. The hash leaf n_i was stored as a leaf-preimage tree as shown in Figure 6.

Algorithm 1. Merkle Tree

Hash each digital signature d, construct Merkle tree and compute the Merkle root M_{r_i}

1. Read digital signature d
2. For all digital signature
3. $n_i = \text{SHA256}(d_i)$
4. Sort all the hashes
5. If $length(n_i)\ mod\ 2 \neq 1$ then
6. loop $i = 1$ to $length(n_i)\ do$
7. $M_{r_i} \leftarrow \text{SHA256}(n_i + n_{i+1})$
8. else
9. If ! last$(length(n_i))$ then
10. loop $i = 1$ to $length(n_i)\ do$
11. $M_{r_i} \leftarrow \text{SHA256}(n_i + n_{i+1})$
12. else
13. $M_{r_i} \leftarrow \text{SHA256}(n_i + n_i)$
14. end if
15. end if

Each hash of leaf n_i will be hashed with partner leaf until the value of the root M_{r_i} was found. This root value will then be one of the information listed in the block entered in the blockchain as the integrity of transaction data from the actual block. If there was a change in the root value, it is necessary to check whether the root value of the block owned by the owner of the block with the one on the blockchain were equal. If the root value changed, the search to verify the node must be based on the node that has been changed.

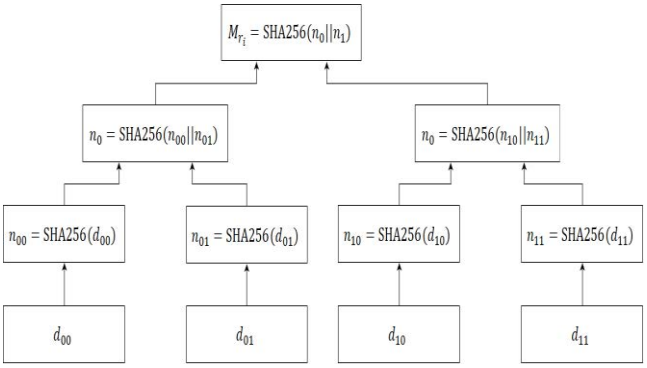

Figure 6. Merkle Tree Scheme

To check whether the node was changed, first assume the block data in node n_{000} was the valid node. The node n_{000} should be checked by using the value of leaf n_{01}, inner leaf n_1, and root M_{r_i}. Furthermore, each known value should be calculate until the hash root was found and check whether the hash root value is now equal to the value of the published root or not. If they are equal then the node n_{000} accepted as authentic. If not, then it was indicated that the signature on node n_{000} had been changed and verify the signature n_{000} should be verified using the signature verification as shown in Section 3.2.

4. EXPERIMENT RESULTS AND ANALYSIS

This section describes the experiment of the security analysis for Merkle tree and the probability of the double spending attack during forking.

4.1.1 Modified Forking Scheme for Decreasing Double Spending Attack in Fawkescoin

The forking protocol scheme showed how the forking occurred in the previous method (as shown in Figure 7) and the proposed method (as shown in Figure 8). In this case, only attacker, owner and attacker network were involved. The fork is declared successful if the ability to mine the blocks on an attacker network is greater than the block owner network. With this capability, attackers could make intentional forks on the blockchain and double spending the transactions in the block.

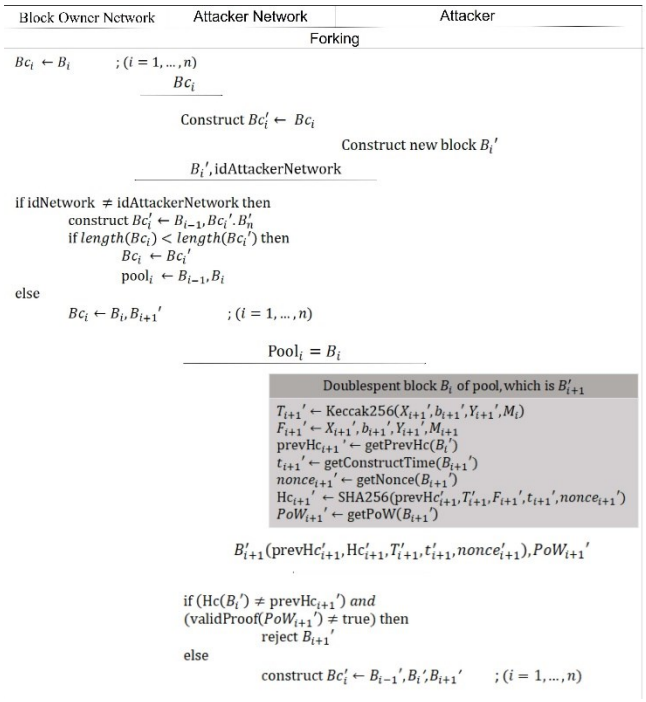

Figure 7. Forking Scheme on Previous Method

To generate forking, the attacker will make a counterfeit chain as shown in Figure 7 and Figure 8. In this case, the attacker will construct the counterfeit blockchain Bc_i' longer than the valid blockchain Bc_i by adding a new B_i' block into Bc_i'. But if the Bc_i chain is longer than Bc_i', then the attacker must improve computing ability if he continued to use the counterfeit chain Bc_i'. If Bc_i' has taken over Bc_i, then the block B_i (the orphaned block) was overwritten by the block B_i' and it was stored in the pool that contains a queued of transactions. As the impact the transaction on the block can be double spend by the attacker. The probability of successful forking [9] was explained as follows.

Proposition 1. *Let the probability of hash rate block owner and hash rate attackers to construct the next block that is* $p = (1 - q)$ *and* q. *The attacker is successful for forking if the attacker builds a counterfeit chain that is longer than a valid chain. The probability* P_{Bc_i}' *of a successful attack on* Bc_i *as shown in Eq.8.*

$$P_{Bc_i}'(B_i, B_{i+1}) = P(B_i' \geq B_i - 1, B_i' + B_{i+1}' > B_i + B_{i+1}) + P(B_i' < B_i - 1, B_{i+1}' > B_{i+1}) \quad (8)$$

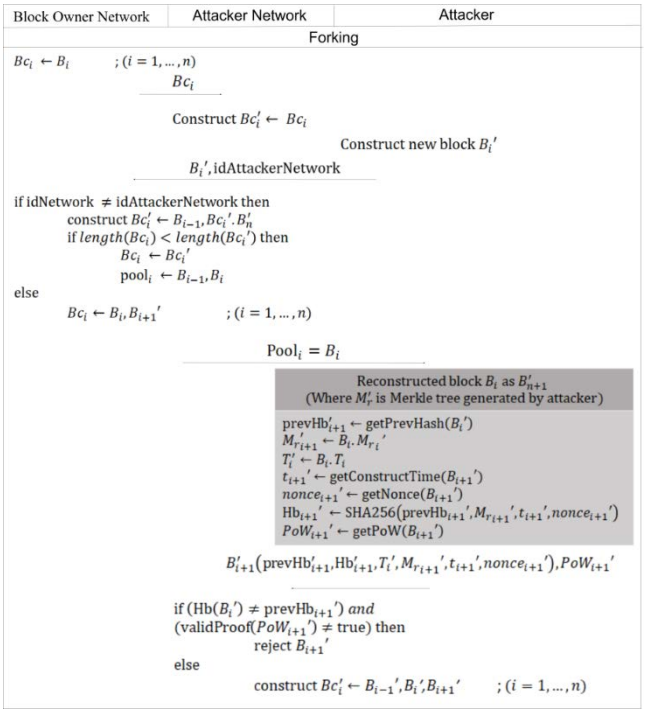

Figure 8. Forking Scheme on Modified Fawkescoin using Proposed Method

Where $P(B_i' \geq B_i - 1, B_i' + B_{i+1}' > B_i + B_{i+1})$ is the joint probability of $B_i' \geq B_i - 1, B_i'$ and $B_i' + B_{i+1}' > B_i + B_{i+1}$,while $P(B_i' < B_i - 1, B_{i+1}' > B_{i+1})$ is the joint probability of $B_i' < B_i - 1$ and $B_{i+1}' > B_{i+1}$. Let $P(i)$ as the probability that the attacker construct a counterfeit chain that is longer than a valid chain, when a counterfeit chain is i blocks behind the valid chain as follows the Eq. 9.

$$P(i) = \begin{cases} 1 & if\ q \geq p \\ a_i, & if\ q < p, \end{cases} \qquad i = 0,1,2, \dots \tag{9}$$

If $q \geq p$, the attacker immediately took the blockchain with probability 1. Otherwise, then the attacker took over the blockchain with the probability of a_i. When the number of block $i \geq 0$, then to find a_i, Eq.10 was used.

$$a_i = (\frac{q}{p})^{i+1}, \qquad i = 0,1,2, \dots \tag{10}$$

$Proof$. Note that a_i satisfies the recurrence relation as shown in Eq.11.

$$a_i = pa_{i+1} + qa_{i-1} \tag{11}$$

Where a_{i+1} is the conditional probability p that the block owner could successfully find the next block and a_{i-1} is the conditional probability q that the attacker can successfully find the next block.

Assumption 1. *We assume that the number of successes block m found by the attacker, with the probability q before the block owner found the block n. The probability successes block m as shown in Eq.12.*

$$P(m,n) = \binom{m+n-1}{m} p^n q^m \tag{12}$$

Where $m < n$, with $m = 0,1,2, \dots$ and $n = m, \dots, \infty$. After block n is found by the actual block owner when the attacker has found the m + 1 block, the race will begin to find a longer blockchain. It follows the probability that the attacker is successful in attacking the blockchain, as shown in Eq.13.

$$P(m,n) = \sum_{f=0}^{\infty} P(m,n) a_{(n-m)} \tag{13}$$

Then, using Eq. 13 with substituting Eq. 11 and Eq.12, the probability to successful attack on the blockchain was calculate using Eq. 14.

$$P'_{Bc_i} = \sum_{B_i'=B_i-1}^{\infty} \sum_{B_{i+1}'=0}^{\infty} P(B_i', B_i) P(B_{i+1}', B_{i+1}) a_{((B_i+B_{i+1})-(B_i'+B_{i+1}'))} + \sum_{B_i'=0}^{B_i-1} \sum_{B_{i+1}'=0}^{\infty} P(B_i', B_i) P(B_{i+1}', B_{i+1}) a_{(B_i-B_{i+1}')} \tag{14}$$

The results of the double spending probability that are successful during the fork is shown in Figure 9. In the previous method [3] ([2]) the double spend attack will be succeed in each block B_i, such that the probability of the attacker succeeding the double spending attack is 1 with the hash rate q and block confirmation number n. Meanwhile, in the proposed method, the possibility of a double spending attack during forking is lower than the probability using the previous method. Significantly successful attack occured if the attacker has a hash level q greater than 50% of the hash block owner's level with different number of block confirmations. Based on Figure 9, it is proven that the proposed method, can reduce the probability of successed double spending attack better when using the previous one. To prevent further attacks the Merkle tree was introduced in the block B_i for preserving the data integrity and securing the transaction. Merkle trees was sending only the root hash in the blockchain. However, if the attacker knows and wants to change the nodes, signature verification would be applied, so that the attacker cannot arbitrarily change and double the transaction spent.

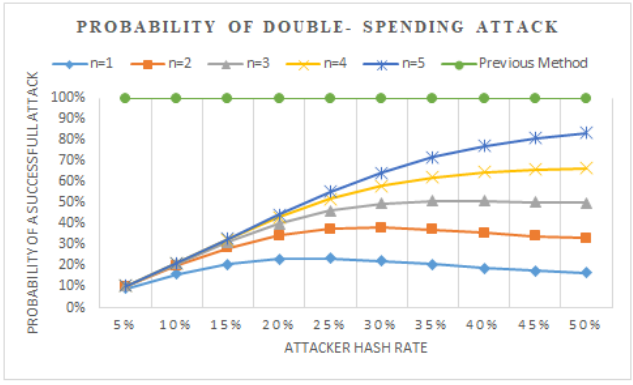

Figure 9. Probability of Successful Double Spending Attack during Forking with Attacker Hash Rate q, for different value of the number of confirmation n.

4.1.2 Merkle Tree for Fawkescoin

Merkle trees are used as data integrity in the blockchain, therefore Merkle tree must be resistant to the possible attacks on each input node, such as collision attacks. Merkle tree with height h leaves has a binding commitment if the underlying hash function H is collision resistant [10].

Proposition 2. *Let hash function* $H : \{0,1\}^{2n} \rightarrow \{0,1\}^n$, *with the height h. For n-bit block* $n_1, \dots, n_{2^h}$ *we define the hash leaf with partner node* $H(H(d_i), H(d_{i+1}))$, *where* d_i *and* d_{i+1} *are the value of digital signature. Then, the root value that contain hash leaf partner of n is* $Mr_i = H(n_i, n_{i+1})$ *where* $H' : \{0,1\}^{2n} \rightarrow \{0,1\}^n$. *The attacker output root such that* $H(n_i, n_{i+1}) = H'(n_i', n_{i+1}')$, *with* $n_i, n_{i+1} \neq n_i', n_{i+1}'$.

Assumption 2. *We assume the attacker occur on the collision resistance of H', with some probability* $P(q)$.

The probability of finding tree hashes collide at the root, is shown in Eq.8

$$P'_{Mr} = P(n_i, n_{i+1} \neq n_i', n_{i+1}') \wedge P\big(H(n_i, n_{i+1}) = H(n_i', n_{i+1}')\big) \quad (8)$$

By applying Eq.8 recursively the hash collision resistance and all identical node would be found with the probability $P(q = 1) \geq P(p = 1)$.

To avoid the attack, the Merkle tree requires logarithmic space and time. The computation of space capacity using the proposed algorithm is less than 3 $log\ n$ and time is $log\ n$. Therefore, the time complexity needed to construct a Merkle tree with height h is $O(log\ n)$. Thus, compare with the previous method which has time complexity $O\ (n)$, the proposed method is better.

5. CONCLUSIONS

In this paper, we present the final message in fawkescoin using DSA and Merkle tree to strengthen the security of fawkescoin against attacks of double spending during forking. Based on the analysis the proposed method produces the time complexity $O\ (log\ n)$ and double spending attacks may successed only if the attacker's hash rate greater than 50%. As for if the attacker knows the node on the Merkle tree, the signature verification will be carried out to determine the owner of the legitimate transaction, thus the attacker cannot arbitrarily change and multiply the transactions in the block that are overwritten. Therefore, it can be concluded that the proposed method prevents fawkescoin from double spending attacks better than the previous one.

6. ACKNOWLEDGMENTS

We thank Allah SWT and all those who have supported us.

7. REFERENCES

[1] Okupski, K. (2016). Bitcoin Developer Reference.

[2] Anderson, R., Bergadano, F., Crispo, B., Lee, J.-H., Manifavas, C., & Needam, R. (1998). A New Family of Authentication Protocols.

[3] Bonneau, J., & Miller, A. (2014). Fawkescoin: A cryptocurrency without public-key cryptography. *Security Protocols XXII*.

[4] Castor, A. (2017, May 16). *A Short Guide to Bitcoin Forks*. Retrieved from Coindesk: https://www.coindesk.com/short-guide-bitcoin-forks-explained/

[5] Pérez-Solá, C., Delgado-Segura, S., Navarro-Arribas, G., & Herrera-Joancomartí, J. (2017). Double-spending Prevention for Bitcoin Zero-confirmation Transaction. *IACR Cryptology ePrint Archive*.

[6] Schneier, B. (1996). *Applied Cryptography, Second Edition: Protocols, Algorithm, and Source Code in C.* John Wiley and Sons, Inc.

[7] Hoffstein, J., Pipher, J., & Silverman, J. H. (2000). An Introduction to Mathematical Cryptography.

[8] Ederov, B. (2007). Merkle Tree Traversal Techniques.

[9] Rosenfeld, M. (2012). Analysis of Hashrate-based Double-spending.

[10] Rogaway, P., & Shrimpton, T. (2014). Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance,and Collision Resistance. *Fast Software Encryption(FSE 2004)*.