ISBN: 978-981-18-7950-0
2023 the 13th International Workshop on Computer Science and Engineering (WCSE 2023)
doi: 10.18178/wcse.2023.06.048

Implementation of an Efficient Multi-Signature Scheme for Copyright Protection on Ethereum

Hye-Young Kim[1] +

[1] School of Games, Major of Game Software
Hongik University
Sejong-si, South Korea
hykim@hongik.ac.kr

Abstract. Blockchain has shown great potential in various fields due to its technical advantages such as peer-to-peer, timestamp, consensus algorithm and encryption. In blockchain, the protection method of transaction data or copyright is crucial and cryptographic digital signature technology has been applied as one of the copyright protection methods. The multi-signature scheme provides higher security than single-signature schemes, enhances the transparency of transactions and contracts, and is widely used in distributed systems utilizing distributed ledger technology in blockchain. Multi-signature requires multiple parties to cooperate in order to produce a valid signature, reducing the risk of exposing the entire system to a single point of failure when compared to single-signature schemes. This is an important role in transactions or contracts that require consensus among multiple parties, where each party can sign to implement the agreement, increasing transparency and preventing disputes. However, the cryptographic digital signature is resource consuming and inefficient because the verification of the signature consumes lots of computational resources and excessive number of communications. Therefore, we have proposed an efficient multi-signature scheme based on Schnorr for copyright protection on Ethereum.

Keywords: Schnorr, mult-signature, Ethereum, copyright protection, digital signature

1. Introduction

Since the Ethereum blockchain platform was published, blockchain is studied rapidly to implement and functions that can be applied [1]. In blockchain, the protection method of transaction data or copyright is crucial and cryptographic digital signature technology has been applied as one of the protection methods. Digital signature technology can guarantee the authenticity, integrity, and non-repudiation of data in electronic commerce and electronic government affairs. The signature technique is not affected by the number of signatures, the final result is only one digital signature in length. Signature can mitigate the above deficiencies or bottlenecks faced by the system based on blockchain [2].

A transaction involving multi-partiesneeds signatures to take effect, and it will consume alarge amount of storage on the blockchain. Schnorr signatures feature a rich algebraic structure suitable for the construction of multiparty protocols. Also, it can be further utilized to construct other cryptographic primitives like multi-signatures, threshold signatures, blind signatures, or adaptor signatures, all which have interesting applications especially to lately arising blockchain technologies. Individual cosigners do not reveal their private information during this process, multi-signature schemes allow for the decentralization of the signing process by requiring multiple parties to cooperate in order to produce a valid signature. A trivial multi-signature scheme can be constructed using an arbitrary digital signature scheme i n the following manner. The signers generate their key pairs, individually sign the message, and concatenate their signatures. The verification then proceeds by obtaining all the public keys of the signers and verifying concatenated signatures individually [3]. The length of the multi-signature was reduced to a constant. A single aggregated key can be used to verify the multi-

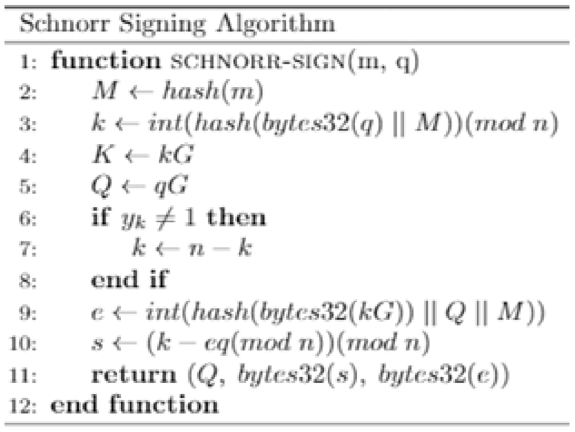
+ Corresponding author. Tel.: +82-10-2310-2710
E-mail address: hykim@hongik.ac.kr

signature. And the individual signatures are bound to the resulting multi-signature; therefore, they cannot be removed. However, the cryptographic digital signature is resource consuming and inefficient because the verification of the signature consumes lots of computational resources and excessive number of communications. Therefore, we have proposed a secure and efficient multi-signature scheme based on Schnorr in this paper.
The rest of this paper is organized as follows. We present the digital cryptographic single and multi-signature algorithms based on Schnorr in section 2. In section 3, we describe the experimental results show that the proposed our multi-signature scheme based on Schorr can effectively improve the performance for copyright protection and reduce communication overhead on Ethereum. Finally, section 4 concludes this paper.

2. Schnorr Signature

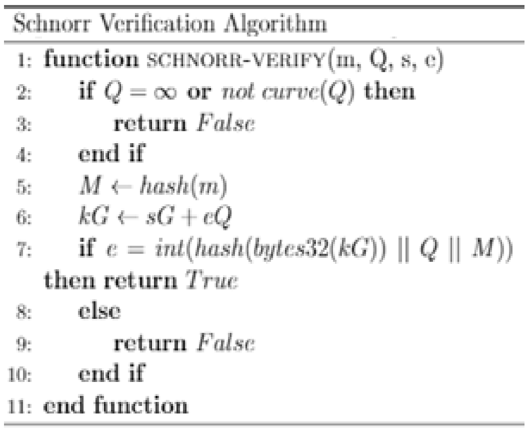
Schnorr signature has a characteristic of fast sign speed and short signature length. It usually uses the P-256 curve released by NIST (national Institute of Standards and Technology) in the United States, and uses sec256r1 curve by SEC. It has been proven strongly unforgeable against adaptively chosen-message attacks in the random oracle model (ROM) [4] by Pointcheval and Stern [5]. We have represented a message, *m* and a private key, *q* as input values in our implementation of signing and verification of Schnorr scheme as in Fig.1 and Fig. 2. In a k value, we hash the private key and the message with the keccak256 hash function and obtain two public keys (x, y) coordinates by multiplying the generator point *G* in *q* by the elliptic curve scalar. Find the equation $e = hash(kG\ |Q\ |\ M)$ required for Schnorr signature and make $s = k - eq$ the equation. The public keys, *s*, and *e* signature values are outputs as return values. It has a message and , public key, Q, and *s*, and *e* as input values that are result values from the signature algorithm in verification algorithm of our scheme. If the public key has an infinite value or does not exist in the elliptic curve, return value is *False*. As long as the equation $kG = sG + eQ$ is performed and the result of hash with the e signature value received as an input value is compared with the equation $e = hash(kG\ |Q\ |\ M)$. If it is wrong return *False*, else if right return *True*. Equation (1-3) performed the verification below.

$$sG + eQ = (k - eq)G + eqG \qquad (1)$$
$$= kG - eQ + eQ \qquad (2)$$
$$= kG = K \qquad (3)$$



```
Schnorr Signing Algorithm
function SCHNORR-SIGN(m, q)
    $M \leftarrow hash(m)$
    $k \leftarrow int(hash(bytes32(q) \,||\, M))(mod\ n)$
    $K \leftarrow kG$
    $Q \leftarrow qG$
    if $y_k \neq 1$ then
        $k \leftarrow n - k$
    end if
    $e \leftarrow int(hash(bytes32(kG)) \,||\, Q \,||\, M))$
    $s \leftarrow (k - eq(mod\ n))(mod\ n)$
    return $(Q,\ bytes32(s),\ bytes32(e))$
end function
```

Fig. 1: Schnorr signing algorithm



```
Schnorr Verification Algorithm
function SCHNORR-VERIFY(m, Q, s, e)
    if $Q = \infty$ or $not\ curve(Q)$ then
        return $False$
    end if
    $M \leftarrow hash(m)$
    $kG \leftarrow sG + eQ$
    if $e = int(hash(bytes32(kG)) \,||\, Q \,||\, M))$
then return $True$
    else
        return $False$
    end if
end function
```

Fig. 2 Schnorr verification algorithm

Due to the characteristic property of SMPC (Secure Multi-Party Computation), individual cosigners do not disclose their private information [6] and the signers generate their key pairs, individually sign the message, and concatenate their signatures. The verification proceeds by obtaining public keys of all the signers and verifying concatenated signatures individually [7]. Every single transaction thathappened on the network will be signed and recorded in the blockchain. A transaction involving multi-partiesneeds multiple signatures to take effect, and it will consume alarge amount of storage if everyone signs the transaction andstores the record on the blockchain. Every node on the blockchain keeps the same ledger, records need tobe as compact as possible.

So, the multi-signature technique was applied to reduce bandwidth and storage volume in some services. The muti-signature technique based on Schnorr scheme is especially attractive for blockchain where communication and storage are more expensive than computation. It produces a multi-signature of constant length instead of linear i n the number of signers. The advantage of having multi-signature of constant length, other than the apparent implications of smaller size, is faster multi-signature verification. The multi-signature verification time can become even constant if another requirement, described i n the following paragraph, is satisfied.

Our proposed multi-signature singing scheme has described in pseudocode in Fig. 3. We have represented *HASH-CAL*( ) for computing communication and *HASH_AGGR*( ) for computing coefficients in key aggregation phase and *HASH-CAL*( ) for used to compute a challenge in the signing procedure of hash function. A random private key Q_i *is* sampled from $\mathbb{Q}_i$, and the group generator *G* is multiplied by this private key to obtain Q_i = q_iG. The resulting key pair is (q_i, Q_i). Each key is prior to that multiplied by a factor ai which is dependent on the whole set of public keys of signers and also on the multiplied key itself. The factors ai are computed as ai = *HASH_AGGR*(P_i, Q_i), and the group key is then the sum $Q = \sum_{i=1}^{n} a_i Q_i$. Each signer *i* initiates the algorithm with a multi-set of public keys $P=\{Q_1, Q_2, .., Q_n\}$ and its private key Q_j. The algorithm then computes a, of the specific signer and the aggregated group key *Q = MERGER(P)*. The participants sample a nonce r_i from $\mathbb{Q}_i$ and multiply the group generator by this number to obtain $Q_i = r_iG$. Once they have Q_i, each signer computes its commitment t_i = *HASH-CAL*(Q_i). The commitments are distributed to other cosigners. Upon receiving all commitments t_j, signers distribute theirs Q_i to other cosigners. The signers have to compute *e = SCHNORR-SIGN(m, q)* and their partial signatures $s_i = q_i + ea_iq_i \ (mod\ n)$**.** The partial signatures are then distributed to other cosigners, and the participants compute $s = \sum_{i=1}^{n} s_i (mod\ n)$. The resulting multi-signature is the pair *(Q, s)*. As shown in Fig. 4, the verifier computes *e = SCHNORR-SIGN(m, q))* and if the equation *sG = Q + eQ* holds, the signature is valid, otherwise, it is invalid.

```
Schnorr Multi-Signing Algorithm
1: function SCHNORR-MULTI-SIGN(m, P, q(i))
2:     a(i) = HASH-AGGR((P), Q(i))
3:     Q = MERGER(P)
4:     k(i) ← Q(q)
5:     Q(i) = r(i)G
6:     t(i) = HASH-CAL(Q(i))
7:     send t(i) to other cosigners
8:     send Q(i) to other cosigners
9:     if HASH-CAL(Q(i)) ≠ t(i)
10:        break
11:    for i = 1 to n
12:        Q = Q +Q(i)
13:    e = HASH_SIGN(m, Q)
14:    s(i) = r(i) + ea(i)q(i) (mod n)
15:    send s() to other cosigners
16:    for i = 1 to n
17:        s = s + s(i) (mod n)
18:    return(Q, bytes32(s))
19: end function
```

Fig. 3: Multi-Signature singing algorithm

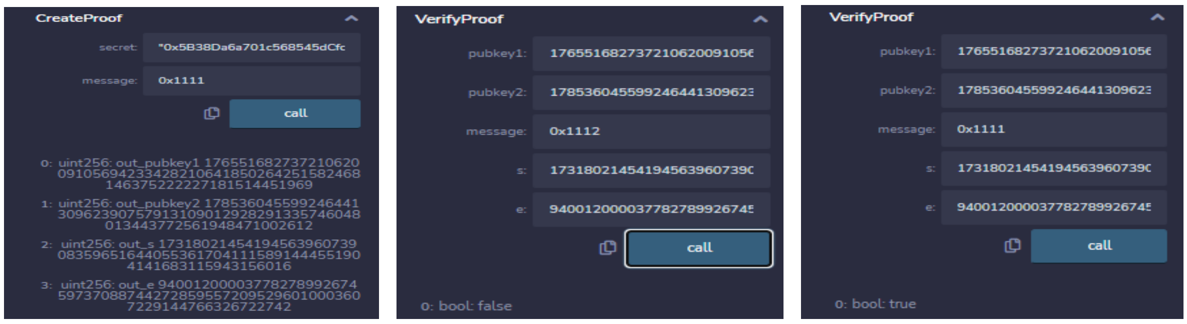
```
Schnorr Multi-Verification Algorithm
1: function SCHNORR-MULTI-VERIFY(m, Q, s)
2:     e = HASH_SIGN(m, Q)
3:     if  sG = Q ⊕ eQ
4:         return True
5:     else
6:         return Flase
7: end function
```

Fig. 4: Multi-signature verification algorithm

3. Simulation Results

Ethereum can access the main network, test network, and private network through the Go-Ethereum (Geth) client. Geth [8] is a client that allows access to the Ethereum blockchain network and is written in the Go language. We have used the so-called Ganache, which corresponds to the private network.

Solidity language [8] was designed by the Ethereum Solidity team, starting with Gavin Wood's proposal in August 2014, to implement Ethereum smart contract functionality operating on EVM(Ethereum Virtual Machine) in a static type of programming language. Solidity language is compiled into byte codes that can operate on EVMs, and developers can implement applications by putting their own business logic into smart contracts through Solidity language. Items recorded in smart contracts cannot be denied and cannot be modified once deployed, so they must be carefully written. Remix IDE [9] is an integrated development environment site that enables integrated coding, compilation, debugging, and deployment of Solidity languages on a browser. It can be used in a web browser without a separate installation, has its own compiler 'solc', and can access Ethereum's network. As shown in Fig. 5, secret means a private key, and message can be viewed as the value to be signed. It can be seen that two public key values and *s* and *e* values, which are signatures required, appear as result values. The SCHNORR-VERIFY function is an algorithm that verifies *s*, *e* signatures, public keys, and messages as input values in the Schnorr signature algorithm.



```
CreateProof
secret: "0x5B38Da6a701c568545dCfc
message: 0x1111
call

0: uint256: out_pubkey1 176551682737210620
   091056942334282106418502642515824681
   46375222227181514451969
1: uint256: out_pubkey2 178536045599246441
   309623907579131090129282913357460480
   13443772561948471002612
2: uint256: out_s 173180214541945639607390
   835965164405536170411158914445519041
   41683115943156016
3: uint256: out_e 940012000037782789926745
   973708874427285955720952960100036072
   29144766326722742
```

```
VerifyProof
pubkey1: 17655168273721062009105€
pubkey2: 17853604559924644130962€
message: 0x1112
s: 173180214541945639607390
e: 94001200003778278992674€
call

0: bool: false
```

```
VerifyProof
pubkey1: 17655168273721062009105€
pubkey2: 17853604559924644130962€
message: 0x1111
s: 173180214541945639607390
e: 94001200003778278992674€
call

0: bool: true
```

Fig. 5: Verification of Schnorr signature (false, true)

In the case that the number of signers is 5 and the random number has been generated, the comparison of the operation of single and multi-signature based on Schorr scheme have shown in Fig. 6. We found that multi-signature scheme is much more efficient in both signing and verification than single signature based on Schnorr scheme on the same situations. The signature length of 2048 bits will also significantly increase the overhead of system operation and storage. As shown in Fig. 7, we have compared the running time of the single and multi-signature based on Schorr scheme when the number of signers is five. The execution time of verification algorithmis not related to the number of signers but the length of the message, so it can be reduced the execution time by performing abstraction for large files, and our multi-signature scheme has shown high efficiency and it will be expected to be applied efficiently in copyright protection on Ethereum.

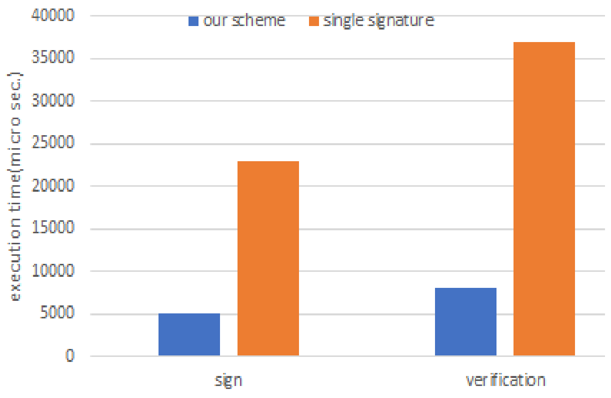


Fig. 6: execution time of signing and verification

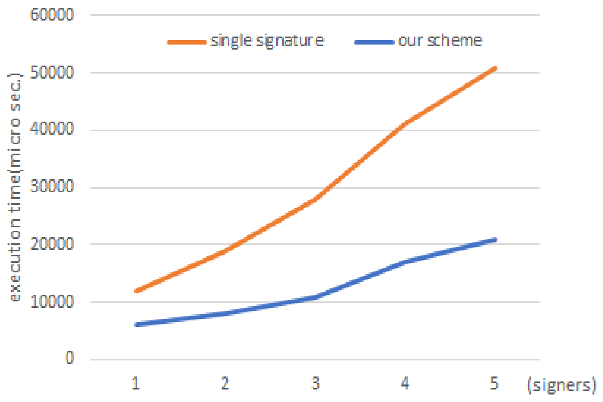


Fig. 7: execution time of multi-signers

4. Conclusion

In this paper, we have proposed an efficient multi-signature scheme based on Schnorr for copyright protection on Ethereum. We describe the experimental results show that the proposed our multi-signature scheme based on Schorr can effectively improve the performance for copyright protection and reduce communication overhead on Ethereum. Also, we have compared the running time of the single and multi-signature based on Schorr scheme when the number of signers is five. We have obtained the result that multi-signature scheme is much more efficient in both signing and verification than single signature based on Schnorr scheme on the same situations. Based on the simulation results in Section 3, our scheme is expected to be applied efficiently in copyright protection on Ethereum.

5. Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2022R1H1A2092553), and this work was supported by the 2023 Hongik University Research Fund.

6. References

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. https://bitcoin.org/bitcoin.pdf

[2] P. J. Taylor, T. Dargahi, A. Dehghantanha, R. M. Parizi and K. R. Choo, “A systematic literature review of blockchain cyber security,” Digital Communications and Networks, vol. 6, no. 2, pp. 147–156, 2020.

[3] Mihir Bellare and Gregory Neven. "Multi-Signatures i n the P l a in Public-Key M o d e l and a General Forking Lemma", In: Proceedings of the 13th ACM Conference on Computer and Communications Security. A C M . 2006, pp. 390-399.

[4] Mihir Bellare and P h i l l ip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: Proceedings of the 1st ACM conference on Computer and communications security. 1993, p p . 62-73

[5] David Pointcheval and Jacques Stern. "Security Arguments for Digital Signatures and B l i n d Signatures". In: Journal of cryptology 13.3 (2000), pp. 361-396.

[6] *Oded Goldreich. "Secure Multi-Party Computation". In: Manuscript. Preliminary version* 78 (1998).

[7] Mihir Bellare and Gregory Neven. "Multi-Signatures i n the P l a in Public-Key M o d e l and a General Forking Lemma", *In: Proceedings of the 13th ACM Conference on Computer and Communications Security.* A C M . 2006, pp. 390-399.

[8] G. Soldati, “An Advanced Signature Scheme: Schnorr Algorithm and its Benefits to the Bitcoin Ecosystem,” Master’s thesis, *Politecnico di Milano*, Italy, 2018.

[9] Remix IDE [Internet], HashNet.