An Efficient Multi-Signature Scheme for Blockchain

Mostefa Kara, Abdelkader Laouid, Mohammad Hammoudeh

[a]*LIAP Laboratory, University of El Oued, PO Box 789, El Oued, 39000, El Oued, Algeria*
[b]*LIAP Laboratory, University of El Oued, PO Box 789, El Oued, 39000, El Oued, Algeria*
[c]*Information & Computer Science Department, King Fahd University of Petroleum & Minerals, Academic Belt Road, 31261, Dhahran, Saudi Arabia*

Abstract

Blockchain is a newly emerging technology, however, it has proven effective in many applications because it provides multiple advantages, mainly as it represents a trust system in which data is encrypted in a way that cannot be tampered with or forged. Because it contains many details such as smart contracts, consensus, authentication, etc. the blockchain is a fertile ground for researchers where they can continually improve previous versions of these concepts. This paper introduces a new multi-signature scheme based on RSA. This scheme is designed to reduce the blockchain's size and prevent known attacks and is also applicable in many other settings that require multi-signatures. Our scheme is in the plain public key model, which means nodes do not need to prove knowledge or possession of their private key. In which, whatever the number of signers, the final signature size is equal to $O(k)$ where k is a security parameter and no interaction between signers is needed. To verify that a number of parties have signed a shared message m, a verifier needs the signature, list of signers, and the message m. The presented practical short accountable-subgroup multi-signature (ASM) scheme allows a valid signature to disclose which subset generated the signature. It is worth noting that our multi-signatures with public key aggregation is an interactive two-round protocol and a multi-signature model applied to the entire block and not to individual transactions.

Keywords: Multi-party computation, Digital signatures, Blockchain, Privacy-preserving, Secure architecture

1. Introduction

As one of the essential elements in the field of information security, encryption plays a central role in all security aspects Zhao et al. (2019). It provides the main foundations for data confidentiality, integrity, and data authentication Chait et al. (2021); KARA et al. (2023). Among the most secure encryption processes currently are Secure Multiparty Computing (SMPC); SMPC allows a set of distributed elements to jointly compute a random function without revealing their input Makri et al. (2021). In recent years, the continuous development of emerging technologies such as cloud computing, blockchain, and the Internet of Things has led to a focus on SMPC. Indeed, SMPC has a unique advantage in solving privacy and security issues, especially since it is a general-purpose tool for calculating private data.

Secure Multiparty Computing (SMPC) is a successful encryption and authentication process in shared computing in a way that maintains the integrity of information collected and stored and preserves confidentiality Kara et al. (2021). In general, the SMPC addresses the problem of collaborative computing performed by a group of participants in a more secure way within the framework of distributed computing Archer et al. (2018). The primary goals that SMPC protocols aim to provide are, first, input confidentiality, as information derived from the implementation of the protocol must not allow any inference as to private data held by others; and second, robustness, which means that no group of conniving elements should be willing to share the information, or any deviation from the instructions is likely to coerce honest elements into producing an incorrect result.

The popularity of blockchain is increasing every day. In which blocks are verified by nodes, which accept them after cryptographic digital signatures. Therefore, an efficient signing process is necessary to ensure that all nodes reach a consensus Habib et al. (2021) and verify the validity of blocks. However, with the widespread use of blockchain, fraudulent activities, such as hacking, have increased. To reduce these fraudulent incidents, multi-signature agreements are effective solutions because they offer an increased level of security.

Protocols that are based on multi-signature are a practice that requires the use of different signatures (hence different keys), rather than just one, to authorize a task He et al. (2021). This practice is of course used to perform authorized party tasks. Today, this technology is widely used in several fields, including electronic transactions. It reinforces the security of transactions in

a more transparent way.

More specifically, a system with multi-signatures requires that several elements sign a transaction before it is integrated into the blockchain Zhang et al. (2022). This approach differs from traditional cryptocurrency transactions, which only require one signature (usually from the sender of funds). These types of systems are sometimes referred to as $t - of - N$ transactions, where t represents the required number of signatures and N is the total number of signatures.

Although there is a lot of research on this technology, there is still a need to develop it. This research presents a new model based on the RSA cryptosystem and signature Rivest et al. (1978), but it merges all signatures into one in a unique and smooth way. There are several application areas of the proposed technique, however, we will assign it in a new paradigm for blockchain, where a new block is only added after being approved by t element, usually, t is equal to 50% of the total system items.

2. Related work

Multi-signatures notion was first presented by Itakura and Nakamura Itakura and Nakamura (1983). Recently, this technology has been investigated broadly based on systems such as RSA, Discrete Logarithms, Pairings, Lattices, etc. In fact, these works vary in improving multi-signature schemes on different levels such as complexity, but they may overlook some other aspects.

In Boneh et al. (2018), the authors constructed multi-signature schemes conceived to reduce the size of the Bitcoin blockchain especially. The proposed constructions in the plain public key model are derived from Schnorr and BLS signatures. To prevent the rogue public key attack, these constructions are based on the techniques invented in Bellare and Neven (2006); Maxwell et al. (2019) for securing Schnorr multi-signatures. This technique still needs three rounds.

The authors in Boneh (2003) presented a concept of an aggregate signature that resembles the multi signatures. In the first one, each element individually creates her own signature, then an aggregator aggregates all generated signatures into one signature. In the second strategy, the signers cooperate to create one signature for the same message. The difference between the aggregate signatures and multi-signatures can be defined not by

the messaging flexibility but by the entity that joins all generated signatures into one.

To strengthen the security of secure multi-party computation (SMC), Blanton et al. Blanton and Jeong (2018) combined SMC schemes based on secret sharing with signatures to enforce input correctness in the form of certification. Based on two types of signatures in the context CL-signatures Camenisch and Lysyanskaya (2004) and ElGamal (1985), they studied the enforcement of truthful inputs used in SMC through input certification at the moment of computation initiation, a party supplying input accompanying it with a certificate, and proves that the data input used in the computation is equivalent to what has been certified.

Pixel Drijvers et al. (2020) is a pairing-based secure multi-signature technique destined for use in blockchains. To protect blockchain against corruption attacks, this protocol allows signers to evolve their keys over time, where new keys cannot be used to sign on old data. However, the calculations are still quite large. In Lee and Kim (2022), the authors proposed a two-round Multi-signature technique based on Okamoto signatures that supports the public key aggregation from a public-key signature technique.

Nurzhan et al. Aitzhan and Svetinovic (2016) handled providing transaction security issues in decentralized smart grid energy trading without dependence on trusted third parties. Using blockchain technology and multi-signatures, they implemented a proof-of-concept (PoC) for a decentralized energy trading system, allowing elements to anonymously negotiate energy prices and securely execute trading transactions. Paper Damgård et al. (2022) proposed many lattice-based distributed signing techniques with low round complexity following the Fiat–Shamir with Aborts (FSwA) paradigm of Lyubashevsky Matsui (2009). These schemes are distributed variants of the fast Dilithium-G signature protocol.

Paper Choi and Kim (2016) presented a lattice-based linearly homomorphic signature in order to have multiple signers with security proof. To distribute different secret keys to each user, the authors used well-known lattice-based protocols such as trapdoor generation and extracting basis.

3. Proposed architecture

In current practice, MultiSign is used for each transaction separately, which undoubtedly requires a lot of time and space to complete the construction of one block. In fact, many techniques have been proposed to solve

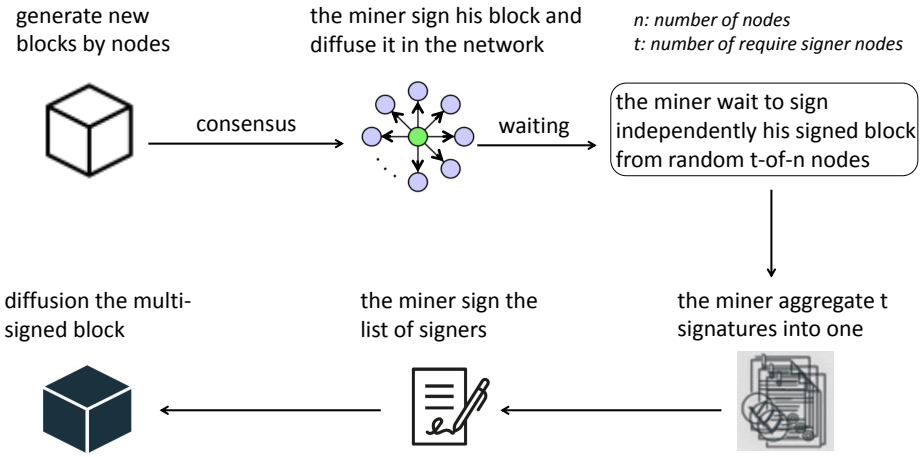

Figure 1: Proposed multi-signature architecture for blockchain

the problem of large space required by public keys, but there are still many signatures in the $t - of - n$ model where n is the total number of signatures and t is the required number of signatures.

In the proposed architecture (Figure 1), the operation is initially normal, i.e., a transaction is signed only by the sender of funds. After creating a new block and going through the consensus process, the miner signs their own block (hash of that block) and then broadcasts it to the network. Each node receives this signed hash, signs it again (signature of signature), and sends it again to the miner. The miner waits until receiving t signatures on its signed hash. The process here is not subject to arrangement, i.e., doesn't matter the order of receipt of signatures from other nodes, because each node signs the signed hash and then sends it to the miner completely independent of other nodes, so there is no interaction between the signing nodes. Upon receipt of t responses, the miner signs the list of nodes participating in this process and broadcasts it on the network. Any element can extract this list, validate signatures and ensure that a node participates in the process.

The proposed multi-signature technique can be shown in Algorithm 1.

Algorithm 1 MS algorithm

Require: $Block : B, pk : d_0, sk : e_0, modulus : N$
Ensure: $sig : \sigma, signer'slist : SL$

function MINER SIG
    $s_0 \leftarrow sig_{e_0,N}(hash(B))$
    diffuse s_0
    wait until receive t signatures
    (signature of node i : $sig_i \leftarrow sig_{e_i,N}(s_0)$)
    compute σ : $\sigma \leftarrow Product(s_i)$
    construct signer's list L : $L \leftarrow Concatenate(signer's\ numbers)$
    sign L : $SL \leftarrow sig_{e_0,N}(L)$
    diffuse σ, SL
end function

4. Model description

The proposed multi-signature scheme consists of three algorithms:
$KeyGen$ is an algorithm that, on input public modulus, generates a public-private key pair (pk, sk).
$Sign$ is an algorithm that on input a secret key sk and message m from the message space, outputs signature σ.
$Verify$ is an algorithm that on input a public key pk, a message m, and a signature σ, outputs a bit.

Our multi-signature model for a message space M is based on the RSA cryptosystem. Its detail is as follows:
Setup:
We assume that there is a trusted third party (TTP) where all nodes agree on the public parameters pp. The setup algorithm used by TTP mainly fixes the distribution of the parameters given an RSA modulus (N). The public parameters are taken to be an implicit input to all of the following algorithms.
Key generation:
The TTP runs the key generation algorithm on the same input N to generate a public verification key d and an initial secret signing key e for each node, where:

$$\begin{cases} N = p \times q \text{ with } p \text{ and } q \text{ are two large prime numbers} \\ \phi(N) = (p-1) \times (q-1) \\ e \times d = 1 \mod \phi(N) \end{cases}$$

Therefore, for a node i there are (e_i, d_i) where $(m^{e_i})^{d_i} \mod N = m$

Signing:
In the first round, the miner (we always call it $node_0$) signs his block as fellow:

$$s_0 = (h \times d_0)^{e_0} \tag{1}$$

Where h is the block hash, d_0 is the public key of the miner, and e_0 is its private key.
Then, the miner diffuses s_0 on the network and waits until receiving t response where t is the required number of signatures. Each node receive s_0, sign it as presented in Equation 2

$$s_i = (s_0)^{e_i} \tag{2}$$

where e_i is the private key of node i.
In the second round of communication, a node i sends its signature s_i to the miner. The miner aggregate all received signatures by multiplying them as fellow:

$$\sigma = \prod_{i=1}^{t} s_i \tag{3}$$

Knowing that nodes are numbered from 1 to n and numbers are of the same size (e.g. $node_1 = 0000000001$, $node_{122} = 0001111010$, $node_{564} = 1000110100$, etc. for numbering 1023 nodes), the miner constructs a list of signers as fellow:

$$L = \|_{i=1}^{t} node_i \tag{4}$$

Then, it signs this list using its private key

$$SL = L^{e_0} \tag{5}$$

Finally, the miner diffuses a signed block (SB) on the network where

$$SB = (Block, d_0, \sigma, SL) \tag{6}$$

Verification:
After diffusing SB, anyone can verify the signature according to the following steps:

- 1: calculate h: $h = Hash(Block)$
- 2: extracting the signer's list using miner public key: $L = SL^{d_0}$
- 3: using miner and signers public keys, the verifier calculates s' as indicated in Equation 7

$$s' = \prod_{i=1}^{t} (h \times d_0)^{D_i} \tag{7}$$

  where $D_i = \prod_{j=1}^{t-1} d_j$ with $j \neq i$; i.e., D_i is the product of all signers public keys except i.

- 4: using miner and signers public keys, the verifier calculates s'' as shown in Equation 8

$$s'' = (((\sigma^{d_0})^{d_1})^{d_2})^{\ldots d_t} \tag{8}$$

- 5: if $s' = s''$, the signature is accepted.

Correctness:
We put $\alpha = h \times d_0$
On the one hand,
we have $s' = \prod_{i=1}^{t} (h \times d_0)^{D_i} \Rightarrow$
$s' = \prod_{i=1}^{t} \alpha^{D_i}$ that implies,
$s' = \alpha^{D_1} \times \alpha^{D_2} \times \ldots \alpha^{D_t}$ that we gives,
$s' = \alpha^{d_2 \times d_3 \ldots d_t} \times \alpha^{d_1 \times d_3 \ldots d_t} \times \ldots \alpha^{d_1 \times d_2 \ldots d_{t-1}}$
On the other hand,
we have $\sigma = \prod_{i=1}^{t} s_i$ where $s_i = (s_0)^{e_i}$ with $s_0 = \alpha^{e_0}$
$\sigma = s_1 \times s_2 \times \ldots s_t$
so $\sigma = s_0^{e_1} \times s_0^{e_2} \times \ldots s_0^{e_t}$
and $\sigma = (\alpha^{e_0})^{e_1} \times (\alpha^{e_0})^{e_2} \times \ldots (\alpha^{e_0})^{e_t}$ that implies,
$\sigma = \alpha^{e_0 \times e_1} \times \alpha^{e_0 \times e_2} \times \ldots \alpha^{e_0 \times e_t} \Rightarrow$
we compute s'' where $s'' = (((\sigma^{d_0})^{d_1})^{d_2})^{\ldots d_t}$
knowing that $(\alpha^{e_i})^{d_i} = \alpha \ \forall \ i$,
$s'' = (\alpha^{e_1} \times \alpha^{e_2} \times \ldots \alpha^{e_t})^{d_1})^{d_2})^{\ldots d_t}$ that we gives,
$s'' = \alpha^{d_2 \times d_3 \ldots d_t} \times \alpha^{d_1 \times d_3 \ldots d_t} \times \ldots \alpha^{d_1 \times d_2 \ldots d_{t-1}}$
finally, $s'' = s'$.

5. Experiments and performance

The implementation of the proposed model has been done in Python language running on a personal computer with Processor Intel(R) Core(TM) i3-3110M CPU 2.40 GHz, 2 Core(s), 4 Logical Processor(s), 4 Go RAM.

5.1. Execution time

In our experiments, each node was simulated by a process, where multi-process programming was implemented.

n	t	size(list)	size(N)	sig time	verf time
100	51	357 bits	364 bits	0.055 (s)	0.762 (s)

Table 1: Multi-signature execution time

Table 1 shows the results of executions. n is the number of nodes, and t is the required number of signers. We notice that to encode 100 elements in binary, we need 7 bits. So the list of 51 ($t = 51$) elements requires 357 bits (size(list)); this is why we used a modulus N of size 364 greater than the list size. In the applied experiment, the miner must encode each node number in 7 bits, build a single string (L) by concatenating the 50 codes, and transform this string into a decimal number; using the secret key, the miner encrypts (signs) this number as an SL. In the verification, the verifier decrypts SL using the miner's public key and extracts L, and represents L in binary ($BinaryL$); if size($BinaryL$) is less than 357, it must be completed with '0' on the left. Afterward, the verifier divides $BinaryL$ into 7 bits to identify the signers; finally, uses the public keys of the signers to calculate s' and s''.

5.2. Scalability

From previous experience, we notice that the number of signers (t) is linked to the size of the public key (N) because the numbers of these signers will be concatenated into a single string ($BinaryL$), then write this string as a decimal number ($DicimalL$), then encrypt this number to become (SL). When decrypting (SL) by a verifier to extract the list of signed nodes, it must get the correct value ($DicimalL$). So $DicimalL$ must be less than N. Equation 9 shows the relationship between the number of nodes in the network (n), the number of signers (t), and the public key (N).

$$t \times \ulcorner log_2(n) \urcorner < \ulcorner log_2(N) \urcorner \quad (9)$$

Equation 9 shows that if (n) is a large number then (N) will become a very large number. Therefore, the technique is not scalable. To solve this problem, we suggest selecting a fixed number of signers (e.g. 100) regardless of network size. The members of this list are permanent or variable (randomly) at each consensus iteration or variable after a certain number of iterations. With this solution, adding a block only needs to sign $t-of-s$ nodes instead of $t-of-n$ (where s is the required number of signers) regardless of the total number of nodes (n) within the network.

5.3. Data size

The proposal is a compact multi-signature with public key aggregation where nodes are not aggregated using their public keys but their numbers and this makes a big difference in size. We proposed a short accountable-subgroup multi-signature (ASM) technique. An ASM technique allows any subset s of a set of n elements to sign a message m where a valid signature σ reveals which subset generated the signature. The signature size is only $O(k)$ bits, the aggregate signers' list is only $O(k)$ bits, independent of n, and the signing approach is non-interactive.

In a basic ASM, a signature by a set s is just the concatenation of all the signatures of s elements. For a security parameter k, the aggregate public key size is $O(\mid s \mid \times k)$ bits, and the signature size is $O(\mid s \mid \times k)$ bits. Our ASM scheme rivals the first ASM proposed by (Boneh et al.) where both the signature and the public key size are only $O(k)$ bits beyond the description of the set s, independent of n. To see how all this can be used, consider firstly a Bitcoin $n-of-n$ Multisig address. If each block contains w transactions, the signature size which is currently done in Bitcoin is equal to $O(w \times k \times n)$. The public key size is also $O(w \times k \times n)$. Consider now Boneh et al. Boneh et al. (2018) scheme, both signature size and public key size are $O(w \times k)$ as opposed to $O(k)$ in our technique for both signature and public key size.

Table 2 parameters are according to Boneh scheme Boneh et al. (2018), where there are tx transactions, each containing inp inputs, all from $n-of-n$ multisig wallets. G denotes the space required to represent an element of a group. By selecting the following parameters, $tx = 1500$, $inp = 3$, $n = 3$. For Bitcoin and MuSig Maxwell et al. (2019), $\mid G \mid = 32$B and $\mid Z \mid = 32$B. For Boneh scheme, $\mid G_1 \mid = 96$B, $\mid G_2 \mid = 48$B, and and $\mid Z_q \mid = 32$B. For

our scheme, we assume that the number of nodes is 10^4, and the number of signers s is 50% $(5000-of-10000)$. 10^4 nodes require 14B to be represented, so the list size $\mid L \mid = 5000 \times 14 = 70$KB and $\mid N \mid = 71$KB is enough.

	combined pk size	combined signature size	total size (KB)	threshold support
Bitcoin	$tx \times inp \times n \times \mid G \mid$	$tx \times inp \times n \times 2 \times \mid Z \mid$	1296	linear
MuSig	$tx \times inp \times \mid G \mid$	$tx \times (\mid G \mid + \mid Z \mid)$	240	small
Boneh scheme	$tx \times inp \times \mid G_1 \mid$	$tx \times inp \times (\mid G_1 \mid + \mid G_2 \mid)$	864	any
Ours	$\mid N \mid$	$\mid N \mid$	142	any

Table 2: Comparison of needed space for a block in blockchain Boneh et al. (2018)

6. Security analysis

Since our scheme is based on RSA, we put Lemma 1.

Lemma 1. *If an adversary can compute the private key d with known m^e where $(m^e)^d = m \mod N$, then an RSA-based signature of an arbitrary message m can be forged.*

Proof. If $d' = d \mod \phi(N)$, then $c = m^d \mod = m^{d'} \mod$ and $e \times d' = e \times d = 1 \mod \phi(N)$. So, C is an RSA signature of m.

□

By Lemma 1, the security of our scheme depends on the security of an RSA signature scheme. Since N has two large prime factors p and q, an adversary can not factor it by any integer factoring algorithm. Moreover, $p-1$ and $q-1$ must have large prime factors p' and q', respectively.

7. Conclusion

In this paper, we have introduced a new multi-signature scheme based on RSA encryption-signature. This system is designed to be applied in many settings that require multiple signatures. In our scheme, the nodes do not need to prove knowledge or possession of their secret key. Where, regardless

of the number of nodes in the network, the final signature size in the model is equal to $O(k)$ where k is a security parameter. All interactions between nodes have been removed, except in two rounds between the miner and signers. The proposed architecture is clearly explained and compared to other schemes, including the current application of Bitcoin.

References

Aitzhan, N.Z., Svetinovic, D., 2016. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. IEEE Transactions on Dependable and Secure Computing 15, 840–852.

Archer, D.W., Bogdanov, D., Lindell, Y., Kamm, L., Nielsen, K., Pagter, J.I., Smart, N.P., Wright, R.N., 2018. From keys to databases—real-world applications of secure multi-party computation. The Computer Journal 61, 1749–1771.

Bellare, M., Neven, G., 2006. Multi-signatures in the plain public-key model and a general forking lemma, in: Proceedings of the 13th ACM conference on Computer and communications security, pp. 390–399.

Blanton, M., Jeong, M., 2018. Improved signature schemes for secure multi-party computation with certified inputs, in: European Symposium on Research in Computer Security, Springer. pp. 438–460.

Boneh, D., 2003. Aggregate and verifiability encrypted signature form bilinear maps. Advances in Cryptology-EUROCRYPT 2003 .

Boneh, D., Drijvers, M., Neven, G., 2018. Compact multi-signatures for smaller blockchains, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer. pp. 435–464.

Camenisch, J., Lysyanskaya, A., 2004. Signature schemes and anonymous credentials from bilinear maps, in: Annual international cryptology conference, Springer. pp. 56–72.

Chait, K., Laouid, A., Laouamer, L., Kara, M., 2021. A multi-key based lightweight additive homomorphic encryption scheme, in: 2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), IEEE. pp. 1–6.

Choi, R., Kim, K., 2016. Lattice-based multi-signature with linear homomorphism, in: 2016 Symposium on Cryptography and Information Security (SCIS 2016), The Institute of Electronics, Information and Communication Engineers (IEICE).

Damgård, I., Orlandi, C., Takahashi, A., Tibouchi, M., 2022. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. Journal of Cryptology 35, 1–56.

Drijvers, M., Gorbunov, S., Neven, G., Wee, H., 2020. Pixel: Multi-signatures for consensus, in: 29th USENIX Security Symposium (USENIX Security 20), pp. 2093–2110.

ElGamal, T., 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory 31, 469–472.

Habib, A., Laouid, A., Kara, M., 2021. Secure consensus clock synchronization in wireless sensor networks, in: 2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), IEEE. pp. 1–6.

He, Q., Xin, X., Yang, Q., 2021. Security analysis and improvement of a quantum multi-signature protocol. Quantum Information Processing 20, 1–21.

Itakura, K., Nakamura, K., 1983. A public-key cryptosystem suitable for digital multisignatures. NEC Research & Development , 1–8.

KARA, M., LAOUID, A., BOUNCEUR, A., HAMMOUDEH, M., ALSHAIKH, M., 2023. Perfect confidentiality through unconditionally secure homomorphic encryption using otp with a single pre-shared key. Journal of Information Science and Engineering 39, 183–195.

Kara, M., Laouid, A., Bounceur, A., Hammoudeh, M., Alshaikh, M., Kebache, R., 2021. Semi-decentralized model for drone collaboration on secure measurement of positions, in: The 5th International Conference on Future Networks & Distributed Systems, pp. 64–69.

Lee, K., Kim, H., 2022. Two-round multi-signatures from okamoto signatures. Cryptology ePrint Archive .

Makri, E., Rotaru, D., Vercauteren, F., Wagh, S., 2021. Efficient comparison for secure multi-party computation, in: International Conference on Financial Cryptography and Data Security, Springer. pp. 249–270.

Matsui, M., 2009. Asiacrypt 2009.

Maxwell, G., Poelstra, A., Seurin, Y., Wuille, P., 2019. Simple schnorr multi-signatures with applications to bitcoin. Designs, Codes and Cryptography 87, 2139–2164.

Rivest, R.L., Adleman, L., Dertouzos, M.L., et al., 1978. On data banks and privacy homomorphisms. Foundations of secure computation 4, 169–180.

Zhang, H., Zou, X., Xie, G., Li, Z., 2022. Blockchain multi-signature wallet system, in: Blockchain Technology and Application: 5th CCF China Blockchain Conference, CBCC 2022, Wuxi, China, December 23–25, 2022, Proceedings, Springer Nature. p. 31.

Zhao, C., Zhao, S., Zhao, M., Chen, Z., Gao, C.Z., Li, H., Tan, Y.a., 2019. Secure multi-party computation: theory, practice and applications. Information Sciences 476, 357–372.