IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 September 2023, accepted 1 October 2023, date of publication 5 October 2023, date of current version 11 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3322196

RESEARCH ARTICLE

An Enhanced Threshold RSA-Based Aggregate Signature Scheme to Reduce Blockchain Size

KHALED CHAIT[1], ABDELKADER LAOUID[1], MOSTEFA KARA[1], MOHAMMAD HAMMOUDEH[2], OMAR ALDABBAS[3], AND ABDULLAH T. AL-ESSA[4]

[1]LIAP Laboratory, University of El Oued, El Oued 39000, Algeria
[2]Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia
[3]Faculty of Engineering Technology, Al-Balqa' Applied University, Amman 15008, Jordan
[4]Access Management and Data Protection Division, Information Protection Department, Digital and Information Technology, Saudi Aramco, Dhahran 31311, Saudi Arabia

Corresponding author: Abdelkader Laouid (abdelkader-laouid@univ-eloued.dz)

ABSTRACT The transformative potential of blockchain technology has resulted in its widespread adoption, bringing about numerous advantages such as enhanced data integrity, transparency, and decentralization. Blockchain has effectively proven its ability to establish trustworthy systems across a multitude of applications. As the number of transactions recorded into a blockchain grows, the blockchain's size expands significantly, posing challenges to the network, particularly in terms of storage capacity and processing power. To address this problem, we present a cryptosystem based on RSA to provide aggregate signatures in blockchains. The aggregate signature replaces all transaction signatures of a block. In this scheme, all participating blockchain nodes use the same modulus N, each with its own private and public key pair generated from N. Regardless of the number of transactions, nodes, and signers, the aggregate signature size is always $O(k)$, where k is a security parameter. The miner that constructs a candidate block computes the aggregate signature σ, replaces all transaction signatures by σ, and transmits the block with only one aggregate signature. The proposed scheme incorporates a flexible and accountable subgroup aggregate signature mechanism, allowing any subset t of n total elements to sign data, where t is the required number of signers. To verify that a set of elements signed the block, the verifier requires the aggregate signature, the aggregate public key, and the data hash. This approach requires minimal interaction between the signers, which results in reduced network traffic. Regardless of the network size, there are always $t + n$ exchanged messages. Experimental analysis shows the proposed aggregate signature scheme's effectiveness in increasing security robustness and reducing block size and overall network traffic.

INDEX TERMS Blockchain, digital signatures, aggregate signature, multiparty computation, secure architecture, transaction authorization.

I. INTRODUCTION

The prevailing design of blockchain exposes information to all network participants, making it challenging to maintain privacy and confidentiality [1]. These restrictions become particularly critical in applications involving sensitive data, such as financial transactions or personal information.

The Secure Multiparty Computing (SMC) concept was proposed as a promising solution to address this privacy concern. SMC enables multiple parties to jointly compute a function on their individual private inputs while keeping those inputs secret. Implementing SMC using blockchain, called SMCB, ensures all SMC transactions are recorded as a timestamped source of truth on the blockchain.

One of the key advantages of SMCB is its ability to enable secure data sharing and collaboration among multiple entities without requiring them to trust each other fully. This is achieved via cryptographic protocols, allowing participants to jointly perform computations on their private data without revealing sensitive information to other parties.

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

Consequently, SMCB offers a viable solution for applications where privacy and confidentiality are paramount, such as healthcare, financial services, and supply chain management.

Encryption is a crucial information security aspect in all security services and mechanisms [2]. It provides the main foundations for data confidentiality, integrity, and authentication [3], [4]. Among the most secure encryption methods currently is SMC. SMC allows a set of distributed elements to jointly compute a random function without revealing their input [5]. The continuous development of emerging technologies such as cloud computing, blockchain, and the Internet of Things (IoT) led to an increased focus on SMC. SMC has a crucial advantage in solving privacy and security issues as a general-purpose tool for calculating private data.

SMC is an encryption and authentication protocol in shared computing that aims to maintain the integrity of collected and stored information and preserve confidentiality [6]. In general, SMC addresses the problem of collaborative computing performed by a group of participants more securely within the framework of distributed computing [7]. The primary goals that SMC protocols aim to achieve are, firstly, input confidentiality, as information derived from the implementation of the protocol must not allow any inference as to private data held by others. Secondly, robustness, which means 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 and use of blockchain is increasing every day [8]. 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 [9] and verify the validity of blocks. However, with the widespread use of blockchain, fraudulent activities have increased. Multi-signature agreements and AGgregate Signature (AGS) were introduced to reduce these fraudulent incidents.

SMC properties can be exploited to create an aggregate signature that enables multiple participants to sign a message(s) without revealing their private key. SMC allows a coordinator to create an AGS that can be verified later while keeping private keys safe.

Protocols based on multi-signature and AGS require using different signatures, hence different keys, rather than just one, to authorize a task [10]. This practice is often used to perform authorized party tasks. AGS technology can be widely used in several fields, including electronic transactions. It reinforces the security of transactions more transparently.

More specifically, a system with multi-signatures or AGS requires several elements to sign a transaction before it is integrated into the blockchain [11]. This approach differs from traditional cryptocurrency transactions, which only require one signature, usually taken from the sender of funds. These systems are sometimes called $t - of - n$ transactions, where t represents the required number of signatures, and n is the total number.

After a consensus process and selecting a new block, nodes need to verify all signatures of the block's transactions one by one to be integrated into the blockchain. In the current practice of Bitcoin, multi-sign $t - of - n$ is used for each transaction separately, which undoubtedly requires a big size (global size equals $tx \times t \times |Sig|$ where tx is the number of transactions) and a long time to verify the block validity ($tx \times t$ verifications).

This research presents a new aggregate signature architecture based on a modified RSA [12] cryptosystem (AGS-MR) to minimize block sizes. Our model merges all signatures into a unique one. There are several application areas of the proposed technique. However, this article studies its application to the blockchain.

The scenario presented in Figure 1 is assumed in the general use of the proposed AGS-MR technique. One or more system elements create the data, then published to the network to be signed by a subset of size $t \leqslant n$ and considered approved. An aggregator (can be one of the signers) waits to obtain t signatures, then combines these individual signatures into one. Finally, this aggregate signature is published for verification.

The rest of the paper is organized as follows: Section II discusses and reviews the relevant works. Section III presents some preliminaries for AGS-MR. Section IV explains AGS-MR's architecture. In Section V, the experimental performance analysis of AGS-MR is presented in three main subsections: an analysis of execution time, scalability, and data size. Section VI discusses the security aspect in detail. Lastly, the paper presents conclusions and future work in Section VII.

II. RELATED WORK

Multi-signatures notion was first presented by Itakura and Nakamura [13]. Recently, this technique was investigated broadly based on systems such as RSA, Discrete Logarithms, Pairings, and Lattices. These works vary in improving multi-signature schemes on different levels, e.g., complexity, but they may overlook other aspects.

The authors of [14] presented a concept of an aggregate signature that resembles the multi signatures. In the first one, each element creates its own signature; an aggregator aggregates all generated signatures into one value. 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. The problem with this scheme is that it did not give an aggregate public key. To improve the privacy protection of transaction addresses in the blockchain, Qiao et al. [15] proposed a short signature size aggregation signature scheme, where the aggregate signature size is separated from the number of signers, which reduces the storage overhead. Moreover, the scheme creates a signature

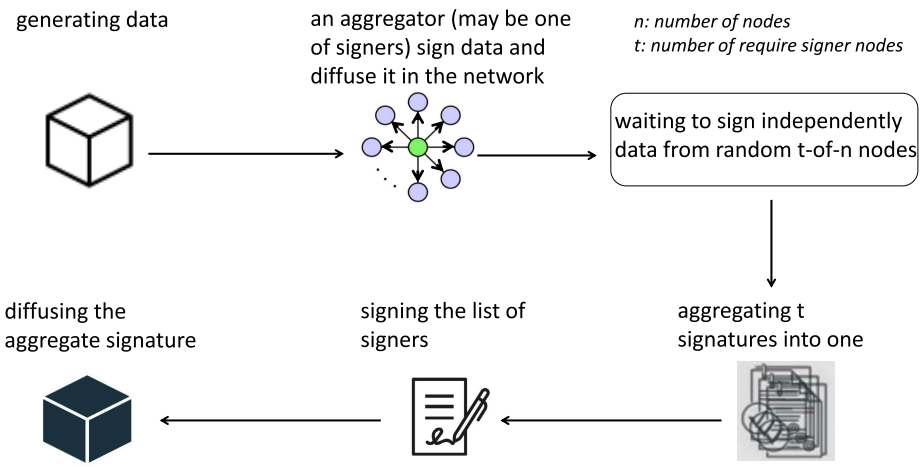

FIGURE 1. Proposed aggregate signature architecture for general use.

based on the discrete logarithm problem, reducing the computational and verification overhead.

In [16], the authors constructed multi-signature schemes to especially reduce the Bitcoin blockchain size. 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 techniques invented in [17] and [18] for securing Schnorr multi-signatures. This technique still needs three rounds. Zhao et al. [19] developed a novel system that combined the general elliptic curve algorithm and Γ−signature schemes in a single scheme as aggregate signatures. First, they proved the subtlety of reaching aggregate signature from general elliptic curves. Then, they proposed an aggregate signature generated from the Γ−signature scheme. Finally, they applied the proposed AGS scheme to the existing Bitcoin system to show the optimized performance and compatibility.

To strengthen the security of secure multi-party computation (SMC), Blanton et al. [20] 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 of CL-signatures [21], [22], 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.

In [23], 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. The signing algorithm takes as input a private key, a list of public keys LK, and a message; thus, each signing process needs a new LK if there is a new signers list. The drawback of this scheme is it needs four exponentiation operations for signing and six for verifying. The authors in [24] developed a secure and feasible permissioned blockchain-based anonymous and traceable aggregate signature scheme for the Industrial Internet of Things. A smart contract is employed to authenticate anonymous sources, smart liaison among entities, and distribute cryptographic materials between entities. Smart contracts ensure data sharing among anonymous and mistrusted entities. The definitive aggregate signature's length is constant, producing an AGS compact.

Nurzhan et al. [25] 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 negotiate energy prices and securely execute trading transactions anonymously. Reference [26] proposed many lattice-based distributed signing techniques with low round complexity following the Fiat-Shamir with Aborts (FSwA) paradigm of Lyubashevsky [27]. These schemes are distributed variants of the fast Dilithium-G signature protocol.

The work in [28] presented a lattice-based linearly homomorphic signature with multiple signers with a security proof. The authors used well-known lattice-based protocols such as trapdoor generation and extracting basis to distribute different secret keys to each user. The problem with this scheme is that the verification process does not give the signers list.

Mihir et al. [29] proposed a two-round multi-signature technique supporting key aggregation to improve the security guarantees for Discrete Log-based multi-signature schemes. The problem is that each individual signature (IS) is sent to every other signer, creating big traffic in the network. In the second round, each signer receives the list of IS and computes the aggregate signature by multiplying them.

Some works on aggregation signature schemes based on blockchain [30], [31], [32], [33]. These proposals suffer from correlation to the individual signature number in the aggregate verification process. In addition, the problem with [30], [31] schemes is the aggregate signature length depends

TABLE 1. A summary of the drawbacks of some previous works.

Scheme	drawback
[14]	it did not give an aggregate public key
[16]	it uses three rounds
[28]	the verification process does not give the signers list
[23]	it needs four exponentiation operations for signing and six for verifying
[29]	each individual signature (IS) is sent to every other signer, so there are $(n \times n)$ messages

on the number of signers. Therefore, they suffer from a big size of AGS.

To address the issues summarised in Table 1, we present a new and effective AGS scheme, which will be described in detail in the next section.

III. PRELIMINARIES

This section recalls some basic definitions and reviews related mathematical concepts.

$\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z} = \{0, 1, \ldots, p-1\}$, denotes a ring of residue classes modulo p,

pk: the public key is a large numerical value that is utilized for encryption and checking the legitimacy of a digital signature.

sk: in a symmetric cryptosystem, the secret key is employed for encryption and decryption; in an asymmetric cryptosystem, the private key is employed for encryption and to construct signatures.

KeyGen: returns a secret (or private) and public keys

mod: modular arithmetic for integers is a function that considers the remainder. In which numbers will wrap around upon reaching a given limit known as the modulus to leave a remainder. This function is often used in prime numbers.

Hashfunction: is a one-way mathematical function that uses inputs of variable lengths and returns outputs of a fixed length. It should be hard to guess the input value using its output. Moreover, it should be hard to determine an input that provides a pre-defined output.

$\times$: multiplication operator,

$+$: addition operator,

$\sum_{i=1}^{t} m_i$: denotes the sum, $m_1 + m_2 + \cdots + m_t$.

$\prod_{i=1}^{t} m_i$: denotes the product, $m_1 \times m_2 \times \cdots \times m_t$

Prime number: a prime number p is a whole number realizes $p \geq 2$, and the only factors of p are 1 and itself.

Safeprimes: A prime number p is called safe if $p = 2 \times p' + 1$, such that p' is also a prime number.

SpecialRSAmodulus: An RSA modulus $N = p \times q$ is special if $p = 2 \times p' + 1$ and $q = 2 \times q' + 1$ are safe primes.

Asymmetricscheme: allows encryption from the recipient's public key. Only the recipient, equipped with a private key associated with his public key, can decrypt the message. The asymmetric technique is based on one-way functions, i.e., it is simple to apply it to plaintext but extremely difficult to recover this plaintext from its corresponding ciphertext without the associated public key.

A. DIGITAL SIGNATURE

Is a mechanism by which a sent message can be authenticated, i.e., verifying that a message comes from a known sender. It must not be easy to forge. Digital signatures are usually made using a hash function and a public-key cryptosystem. The hash provides a message digest, which will be encrypted using a public-key cryptosystem. Therefore, a digital signature is a number dependent on some secret known only to the signer, namely, the signer's private key. Besides, signatures must be easily verifiable.

B. ONE-ROUND PROTOCOL

Supports non-interactive applications like secure e-mail sending. If user A wants to email user B, key k_a is the shared secret key.

C. TWO-ROUND PROTOCOL

Supports interactive applications. If user A wants to communicate with user B interactively, then the keys k_a and k_b are the shared secret keys.

D. THREE-ROUND PROTOCOL

Supports also interactive applications. If user A wants to communicate with user B interactively. This protocol includes key confirmation in the third round.

E. BASICS OF RSA SIGNATURES

Assume a standard RSA setting; the public key is indicated as a pair (e, N), and the private key is indicated as a number d. The modulus N equals a product of two large and secret primes p, q, and the private key d denotes the multiplicative inverse of e modulo $(p-1) \times (q-1)$. To give a secure RSA system, it is assumed that both p and q are sufficiently large, such that it is infeasible to either factorize N or to find the secret d, given only the public key (e, N). Let a message m denote an integer between 0 and N. The RSA signature is constructed as follows: $s = m^d \mod N$. Anyone can verify that s is a signature on the message m as follows: $m = s^e \mod N$.

F. BLOCKCHAIN

Is a new technology for storing and transmitting information evolving without centralized control; it uses peer-to-peer networks and allows the constitution of replicated and distributed registers; these registers are secured thanks to cryptography by linking blocks to each other. It consists of approved transactions before being recorded in a computer code chain. Transaction details are recorded on a public ledger that anyone on the network can see. It can be used in cryptocurrencies, financial contracts, asset tracking, digital Identity, real estate, voting, government benefits, healthcare and medical information, logistics and supply chain tracking, etc.

IV. AGS ARCHITECTURE

This section provides an overview of the proposed AGS-MR, which aims to reduce the size of data represented by a large number of signatures to only one. It also aims to reduce the calculation time by verifying one signature instead of a group of signatures.

We propose a new aggregate signing method based on RSA. Then, we present how to apply it to reduce the size of the blockchain. The same RSA modulus N is used by all participants to enable a coordinator to create an aggregate signature σ based on a set of individual signatures s_i calculated using N. Therefore, it does not matter whether these single signatures belong to the same message or to different messages, as is the case with the blockchain (where the transactions are different). Our approach can be used in: (1) from $(m; s_1, s_2, \ldots s_i)$ to $(m; \sigma)$ where s_i is the signature of user i; Or (2) from $(m_1, m_2, \ldots m_i; s_1, s_2, \ldots s_i)$ to $(m_1, m_2, \ldots m_i; \sigma)$, where s_i is the signature of m_i.

Each node receives from the aggregator the message and its signature, verifies the aggregator signature, signs the message, and sends it back again to the aggregator. The aggregator waits until receiving $t-1$ individual signatures. This process is not subject to arrangement, i.e., the order of receipt of signatures from other nodes is random because each node signs the digest and then sends it to the aggregator completely independent of other nodes. Hence, there is no interaction between the signer nodes. Upon receipt of $t-1$ responses, the aggregator computes the aggregate signature, signs the list of nodes participating in this process, and broadcasts them on the network. Any element can extract the signer list, check the AGS, and easily ensure whether a node participated in the process.

The general architecture of the proposed AGS is shown in Figure 2 and given in Algorithms 3 and 1.

The process in the proposed AGS-MR for blockchain (Figure 3) starts normally, i.e., a transaction is signed only by the sender of funds (or $t-of-n$ multi-sig). After creating a new block and going through the consensus process, the miner signs aggregate tx signatures into one by multiplication. We propose a new modified RSA-based signature using a single common modulus N for all nodes and caching the real encryption key e by a random number r. The AGS is computed by multiplying all individual signatures of transactions.

A. AGS-MR DESCRIPTION

The AGS-MR 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 takes in input the secret key sk and message m from the message space, outputs aggregate signature σ.

Verify: is an algorithm that takes in input the public key pk, a message m, and a signature σ, and outputs a bit '1' denotes accepted, '0' denotes rejected.

Our AGS model for message space M is based on the *RSA* cryptosystem. Its detail is as follows:

1) SETUP

We assume that there is a Trusted Third Party (TTP) where all nodes agree on the public parameters. TTP's setup algorithm mainly fixes the parameters' distribution given an RSA modulus (N). The public parameters are an implicit input to all of the following algorithms.

2) KEY GENERATION

A TTP runs the key generation algorithm on the same input N to generate the node's public and private keys e and d, respectively. Only the TTP knows p and q to prevent any node from generating its own private key. The TTP operates as follows:

$$\begin{cases} get\ N = p \times q\ with\ p\ and\ q\ are\ two\ large\ prime \\ \quad numbers \\ compute\ \phi(N) = (p-1) \times (q-1) \\ select\ at\ random\ e,\ gcd(e, \phi(N)) = 1 \\ compute\ d,\ e \times d \equiv 1 \mod \phi(N) \\ construct(E, D) = \{(e, d)_i, i \geqslant 0\} \end{cases}$$

the subscripts for e and d are as follows:

$$\begin{cases} e\ is\ a\ small\ integer\ random\ number\ :\ 1 < e < \phi(N) \\ d = e^{-1} \mod \phi(N) \\ (m^e)^d \mod N = m \end{cases}$$

The TTP selects a pair from (E, D) we call it (e_r, d_r) where d_r is not prime. Therefore, $d_r = t \times r$ with $(t, r) \in Z^2$. It keeps d_r and r secret, puts t as a threshold, and e_r is a public parameter. Thus, $(m^{e_r})^{t \times r} \mod N = m$.

Finally, the TTP gives each node i a pair of keys (e_i, d_i') where

$$\begin{cases} d_i' = d_i + r \\ (m^{e_i})^{d_i} \mod N = m \end{cases}$$

In another term, d_i is not known for the node i (see Section VI).

3) SIGNING: FIRST ROUND

In the first round, an aggregator $(node_1)$ signs the candidate data as

$$s_1 = h^{d_1'} \tag{1}$$

where h denotes the hash of a message m, d_1' is the aggregator's private key.

Then, the aggregator diffuses s_1 on the network and waits until receiving $t-1$ response, where t is the required number of signatures (the threshold).

Each node receives the signature s_1, checks it using sender's public key (e_1) following these steps:

1) Given $s_1 = h^{d_1'} = h^{d_1 + r}$
2) Let $y = h^{e_r \times t + e_1}$ where e_r is a known public parameter

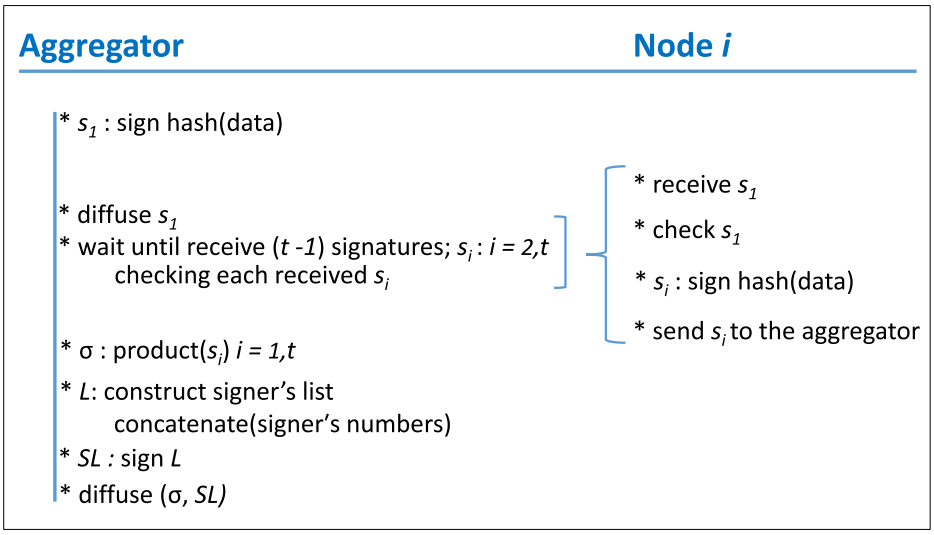

FIGURE 2. Proposed modified RSA-based aggregate signature architecture.

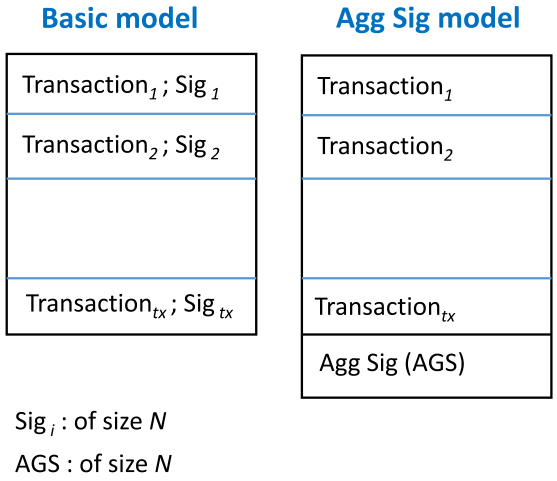

FIGURE 3. Basic and Agg Sig block models.

3) the receiver computes $y' = s_1^{e_r \times t \times e_1}$
4) if $y = y'$, then the sender is checked, and its signature is valid.

4) CORRECTNESS OF A SINGLE SIGNATURE VERIFICATION

$$\begin{aligned} s_1 &= h^{d_1'} = h^{d_1+r}; \\ y' &= s_1^{e_r \times t \times e_1}; \\ \Longrightarrow\ y' &= h^{(d_1+r)\times(e_r \times t \times e_1)} \\ y' &= h^{(d_1 \times e_1 \times e_r \times t)+(r \times t \times e_r \times e_1)}; \end{aligned}$$

we have $d_1 \times e_1 \equiv 1 \mod \phi(N)$ and $r \times t \times e_r = d_r \times e_r \equiv 1 \mod \phi(N)$.

Therefore, $y' = h^{e_r \times t + e_1} = y$.

After checking, each node i signs the message hash using its private key

$$s_i = h^{d_i'} \tag{2}$$

Algorithm 1 Individual Signature Algorithm

Require: $(message : M;\ sig_{aggregator} : s_1;\ pk_{aggregator} : e_1);\ sk : d_i';\ threshold : t;\ pub\ param : e_r, N$
Ensure: $sig : s_i$

function Node$_i$ Sig
  $h \leftarrow hash(M)$
  $y \leftarrow h^{e_r \times t + e_1} \mod N$
  $y' \leftarrow s_1^{e_r \times t \times e_1} \mod N$
  if $y = y'$ **then** ▷ verify aggregator signature validity
    $s_i \leftarrow h^{d_i'} \mod N$ ▷ compute its own signature
    send s_i to the aggregator
  else
    invalid aggregator signature
  end if
end function

where d_i' is the private key of node i.

5) SIGNING: SECOND ROUND

In the second round of communication, a node i sends its signature s_i to the aggregator. The aggregator assembles all received signatures by multiplying them (including its own) as shown in Equation 3.

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

If the list of participants must be shared, we assume that nodes are numbered from 1 to n, and numbers are of the same size (e.g., node 1: n_1 = 0000000001, node 122: n_{122} = 0001111010, node 564: n_{564} = 1000110100, etc. for numbering 1023 nodes), the aggregator constructs list of signers using Equation 4, then signs L using its private

Algorithm 2 Verification of Individual Signature

Require: *message hash* : h; *sig* : s; pk_{sender} : e; *threshold* : t; *pub param* : e_r, N
Ensure: *sig validity* : 0, 1

function Verf Sig
  $y \leftarrow h^{e_r \times t+e} \mod N$
  $y' \leftarrow s^{e_r \times t \times e} \mod N$
  if $y = y'$ **then**
    return 1
  else
    return 0
  end if
end function

key (5).

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

$$SL = (hash(L))^{d_1'} \tag{5}$$

Finally, the aggregator diffuses Signed Message (SM) information on the network where everyone can easily check its validity (6).

$$SM = (m, \sigma, L, SL) \tag{6}$$

6) AGGREGATE SIGNATURE VERIFICATION

Using SM, anyone can verify the aggregate signature by applying the following steps.

1) Calculate h: $h = Hash(message)$.
2) Check the validity of the signer's list by employing Algorithm 2, then extract participating signers.
3) Let $h' = h^{e_r}$.
4) Using aggregator and signers public keys, the verifier calculates s using Equation 7.

$$s = \prod_{i=1}^{t} h'^{E_i} \tag{7}$$

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

5) Using aggregator and signers public keys, the verifier calculates s' using Equation 8.

$$s' = (\sigma^{e_r} \times h^{-1})^{E} \tag{8}$$

where $E = \prod_{i=1}^{t} e_i$ and h^{-1} id the modular multiplicative inverse of h, i.e., $h \times h^{-1} \mod N = 1$.

6) If $s = s'$, the aggregate signature is accepted.

7) CORRECTNESS OF THE AGGREGATE SIGNATURE VERIFICATION

$h' = h^{e_r}$ and $s = \prod_{i=1}^{t} h'^{E_i} \Rightarrow$
$s = h'^{E_1} \times h'^{E_2} \times \ldots h'^{E_t}$. Knowing that $E_i = \prod_{j=1}^{t} e_j$ with $j \neq i$, this gives us,
$s = h'^{e_2 \times e_3 \ldots e_t} \times h'^{e_1 \times e_3 \ldots e_t} \times \ldots h'^{e_1 \times e_2 \times e_4 \times \ldots e_t}$ and

Algorithm 3 AGS Aggregator Algorithm

Require: *message* : M; $pk(s)$: e_i; *sk* : d_1'; *threshold* : t; *pub param* : e_r, N
Ensure: *sig* : σ, *signer's list* : SL

function Aggregator Sig
  $h \leftarrow hash(M)$
3: $s_1 \leftarrow h^{d_1'} \mod N$
  diffuse (M, s_1)
  $list \leftarrow \{\}$
6: $L \leftarrow''$
  while $|list| < t$ **do**
    receive sig s_i of node n_i
9: checking the validity of each s_i
    insert n_i in $list$
    $L \leftarrow L \parallel n_i$
12: **end while**
  $\sigma \leftarrow \prod_{i=1}^{t} s_i \mod N$
  $SL \leftarrow (hash(L))^{d_1'} \mod N$
15: diffuse (M, σ, L, SL)
end function

TABLE 2. Private and public parameters in the modified RSA scheme where $N = p \times q$, $d_r = t \times r$, and $d_i' = d_i + r$.

entity	private	public
TTP	p, q, r, d_r, and d_i	N, t, and e_r
user i	d_i'	e_i

$s = h'^{E_1+E_2+E_3+\ldots E_t}$
On the other hand,
we have $\sigma = \prod_{i=1}^{t} s_i$ where $s_i = h^{d_i'}$ with $d_i' = d_i + r$
$\sigma = s_1 \times s_2 \times \ldots s_t$
so $\sigma = h^{d_1'} \times h^{d_2'} \times \ldots h^{d_t'}$
$\Rightarrow \sigma = h^{d_1+r} \times h^{d_2+r} \times \ldots h^{d_t+r}$ that implies,
$\sigma = h^{t \times r} \times h^{d_1+d_2+\ldots d_t}$
according to Equation 8, $s' = (\sigma^{e_r} \times h^{-1})^{E}$ where $E = \prod_{i=1}^{t} e_i$;
$\Rightarrow s' = (h^{(t \times r) \times e_r} \times h^{(d_1+d_2+\ldots d_t) \times e_r} \times h^{-1})^{E}$
we know that $(t \times r) \times e_r \equiv 1 \mod \phi(N)$ so $h^{(t \times r) \times e_r} = h$; and $h \times h^{-1} = 1$;
that implies $s' = ((h^{e_r})^{(d_1+d_2+\ldots d_t)})^{E}$,
therefore, $s' = (h'^{(d_1+d_2+\ldots d_t)})^{E}$,
we have $E \times (d_1 + d_2 + \ldots d_t) = (e_1 \times e_2 \times \ldots e_t) \times (d_1 + d_2 + \ldots d_t)$,
if $e_i \times d_i \equiv 1 \mod \phi(N)$ then
$E \times (d_1 + d_2 + \ldots d_t) = E_1 + E_2 + \ldots E_t$ and
$s' = h'^{E_1+E_2+\ldots E_t} = s$.
The verification function can be illustrated in Algorithm 4.
Table 2 summarizes the private and public parameters used in the proposed approach.

B. AGS-MR BLOCKCHAIN SCENARIO

In Algorithms 3 and 1, t denotes the number of required signatures. After signing data (data digest) by the aggregator (line 3 in Alg 3), this signature s_1 is diffused (line 4 in Alg 3),

Algorithm 4 AGS Verification Algorithm

Require: $message : M$; $pk_{aggregator} : e_1$; $agg\ sig : \sigma$; $signerslist : L$; $signed\ list : SL$; $pub\ param : t, e_r, N$

Ensure: $sig : s_i$

function Verf Agg Sig
  $h \leftarrow Hash(M)$
  $hl \leftarrow Hash(L)$
  $y \leftarrow hl^{e_r \times t + e_1} \mod N$
  $y' \leftarrow SL^{e_r \times t \times e_1} \mod N$
  if $y = y'$ **then**
    $h' \leftarrow h^{e_r} \mod N$
    $E \leftarrow \sum_{i=1}^{t}(\prod_{j=1}^{t} e_j)$ with $j \neq i$
    $s \leftarrow h'^{E} \mod N$
    $E' \leftarrow \prod_{i=1}^{t} e_i$
    $s' = (\sigma^{e_r} \times h^{-1})^{E'} \mod N$
    if $s = s'$ **then**
      return 1
    else
      return 0
    end if
  else
    return 0
  end if
end function

and each participant node that receives s_1 verifies it (line 5 in Alg 1). If s_1 is valid, the node i signs data (data digest) and sends back its own s_i to the aggregator (lines 6 and 7 in Alg 1). After receiving t signatures, the aggregator computes the aggregate signature σ (line 12 in Alg 3) in the next step. Finally, the list of signers and signs is constructed using Equations 4 and 5.

Where t is the number of block transactions, the scenario in AGS-MR for blockchain differs slightly. Each sender i signs its transaction T_i as shown in Equation 2 where h denotes the transaction hash, i.e., $s_i = (hash(T_i))^{d'_i}$. The miner computes σ the multiplication of all transactions' signatures using Equation 3 where s_i denotes the signature of transaction i. Finally, the miner diffuses the signed block SB: $SB = (block, \sigma)$. The approved block can now be integrated into the blockchain after easily checking its AGS, as shown in Algorithm 4, where the message is the miner's candidate block.

V. PERFORMANCE EXPERIMENTAL ANALYSIS

To assess the performance of the proposed AGS-MR, this section is divided into three main subsections: an analysis of execution time, scalability, and data size.

A. EXECUTION TIME

The proposed AGS-MR was implemented 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), and 4 Go RAM. Each node was simulated by a process where multi-process programming was implemented.

TABLE 3. AGS-MR execution time with different numbers of required signatures.

# signatures	Sig (ms)	Verf (ms)
13	0.06	0.22
29	0.13	0.58
117	0.54	7.39
203	0.93	23.4
351	1.62	74.3
1053	5.04	925

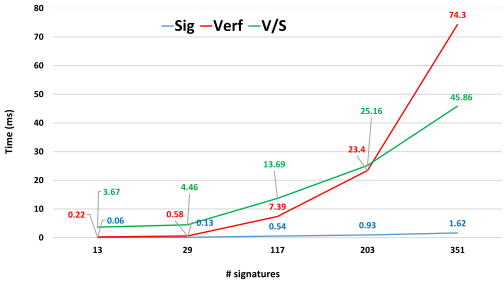

FIGURE 4. Execution time of signing and verifying processes for different numbers of signatures.

Table 3 and Figure 4 show the results of signing and verification functions executions. In these experiments, we ignored the time to build and extract the signers' list in the Sig and Verf phases, respectively. The setting used in the implementation was $N = 32$ bits, t is the required number of signers, and sha256 hash is used. In each test, we chose a secret key d_r, factorized it, and extracted t and r where $d_r = t \times r$.

We notice that to sign 13 transactions, 0.06 *ms* is needed; the verification needs 0.22 *ms*. For 29 transactions, Sig and Verf needed 0.13 *ms* and 0.58 *ms* respectively; this makes sense so that $29 \approx 13 \times 2, 0.13 \approx 0.6 \times 2$ with a little increase in verification time $0.58 \approx 0.22 \times 2.5$. In the third test, the increase rate is fixed for Sig, but it increases for Verf, $117 \approx 29 \times 4$, $0.54 \approx 0.13 \times 4$, and $7.39 \approx 0.58 \times 13$. Comparing tests 5 and 6, $1053 \approx 351 \times 3$, $5.04 \approx 1.62 \times 3$, and $925 \approx 74.3 \times 12$. This is logical because, in the Sig function, we just have a multiplication of signatures; on the other hand, the Verf function uses exponential operations. Figure 5 compares AGS-MR with other techniques in terms of computation cost for aggregate verification.

B. SCALABILITY

Scalability is the extent to which a system accommodates the number of signers. As for the aggregate signature in itself, this does not represent a problem at all because whatever the number of signees, the size of AGS will remain constant (equal to $O(k)$) and will not be affected by that, thus, our proposed AGS is scalable. We will have to discuss the list

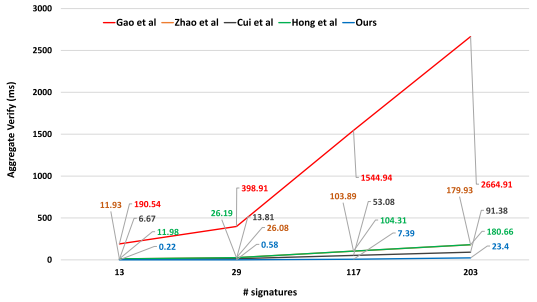

FIGURE 5. The aggregate verification cost comparison.

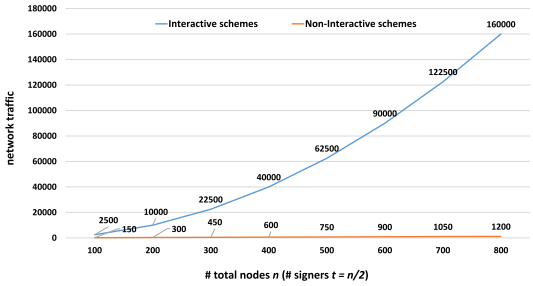

FIGURE 6. Interactive and non-interactive schemes network traffic comparison.

of signers in the threshold system, where we need t signers from n elements.

In the proposed method, a list L of members participating in the aggregate signature is formed as shown in Equation 4, and then this list is sent as it is without being encrypted. To verify the integrity, the aggregator signs the list's digest (SL) using its private key (Equation 5), and everyone can check SL using the aggregator's public key. The size of a list L is not linked to any parameter other than the total number n of nodes within the network. Whatever n, we can control t simply because we can choose a private d_r which can be factorized into several small factors, and then the desired t is controlled.

If t is a large number that will affect the size of the participants' list, we can choose t' signers where $t = t' \times \alpha$. That is, after receiving t signatures, the aggregator randomly divides them into t' groups, where each group consists of α elements. One signer is randomly selected from each group. In this case, it must work with (e'_r, d'_r) instead of (e_r, d_r) where $d'_r = t' \times r$.

In other settings, if we want to encrypt the list of signers, the decimal value of the concatenation of signer numbers must be less than N. The verifier will decrypt this value using the aggregator's public key, knowing that the aggregator's private key encrypts it. Equation 9 shows the relationship between the number of nodes in the network (n), the number of signers (t), and the public key (N).

$$t \times \lceil log_2(n) \rceil < \lceil log_2(N) \rceil \tag{9}$$

where $\lceil x \rceil$ denotes $int(x) + 1$.

Equation 9 indicates that if n is a large number, then N will become a very large one. Therefore, the technique with these settings is not scalable. To solve this problem, we can select a sub-group s of users regardless of network size. The members of s are permanent or randomly variable at each consensus iteration or after certain iterations. Thus, validating a message only needs to sign $t' - of - s$ users instead of $t - of - n$.

C. NETWORK TRAFFIC

Cooperative blockchain techniques, such as multi-signature schemes, can be classified into two primary categories: interactive and non-interactive. Interactive schemes involve complete interaction among all participants to generate a multi-signature or aggregate signature. In contrast, non-interactive schemes minimize system traffic by requiring only partial interaction. For instance, Bellare et al. [29] proposed an interactive scheme that generates $t \times t$ exchanged messages. Conversely, non-interactive schemes, like the proposed AGS-MR, generate only $t + n$ messages (Figure 6), where t represents the number of signers and n represents the total number of nodes (with $t < n$).

D. DATA SIZE

The proposal is a compact AGS with public key aggregation where signers are not aggregated using their public keys but their numbers, which makes a big difference in size. We proposed a short accountable-subgroup AGS technique. An AGS technique allows any subset s of a set S to sign a message m where a valid signature σ reveals which subset generated the signature. The signature size is only $O(k)$ bits independent of t and x, where $t = \mid s \mid$ and $x = \mid S \mid$.

In a basic AGS, a signature by a set S is just the concatenation of all the signatures of x elements. For a security parameter k, the aggregate public key size is $O(x \times k)$ bits, and the signature size is $O(x \times k)$ bits. Our AGS-MR scheme rivals other works where AGS size is only $O(k)$ bits beyond the description of the set S, independent of x. The aggregate public key (AGP) size can be reduced using the order number of elements expressed by b; therefore, AGP size equals $x \times \mid b \mid$ with $b \ll k$.

To see how all this can be used, consider a Bitcoin $n-of-n$ Multisig address first. If each block contains tx transactions, the signature size currently done in Bitcoin is equal to $O(tx \times n \times k)$. The public key size is also $O(tx \times n \times k)$.

Table 4's parameters are set following Boneh scheme [16], where there are tx transactions, each containing inp inputs, all from $n-of-n$ multi-sig wallets. G denotes the space required to represent an element of a group, and "any" denotes support for arbitrary polynomial sizes x and y for $x-of-y$ multi-sig. By selecting the following parameters, $tx = 1500$, $inp = 3$, $n = 3$. For Bitcoin and MuSig [18], $\mid G \mid$ = 32B and $\mid Z \mid$ = 32B. For Boneh scheme, $\mid G_1 \mid$ = 96B, $\mid G_2 \mid$ = 48B, and $\mid Z_q \mid$ = 32B. For our scheme, b = 32B and $\mid N \mid$ = 2KB.

VI. SECURITY ANALYSIS

This Section presents the following points.

TABLE 4. Comparison of needed space for a block in blockchain [16].

	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	$tx \times inp \times \mid G \mid$	$tx \times (\mid G \mid + \mid Z_q \mid)$	240	small
Boneh MSDL	$tx \times inp \times \mid G \mid$	$tx \times (\mid G \mid + \mid Z_q \mid)$	240	small
Boneh MSP	$tx \times inp \times \mid G_2 \mid$	$tx \times \mid G_1 \mid$	360	small
Boneh AMSP	$tx \times inp \times \mid G_2 \mid$	$\mid G_1 \mid$	216	small
Boneh ASM	$tx \times inp \times \mid G_2 \mid$	$tx \times inp \times (\mid G_1 \mid + \mid G_2 \mid)$	864	any
Boneh AASM	$tx \times inp \times \mid G_2 \mid$	$tx \times inp \times \mid G_2 \mid + \mid G_1 \mid$	432	any
Ours	$tx \times inp \times \mid b \mid$	$\mid N \mid$	146	any

A. SECURITY MODEL

TTP selects p and q as two large safe primes numbers where $N = p \times q$ is in order of -at least- 2048 bits and keeps them secret. The TTP selects a secret RSA key pair as (e_r, d_r) where d_r is not prime. Let $d_r = t \times r$. Here, d_r and r are kept secret, and the threshold t is set. The public parameters are (N, t, e_r). Each RSA key pair (e_i, d_i) generated by TTP should be transformed to (e_i, d_i') before it delivers it to the user i, where $d_i' = d_i + r$. The message m that will be signed should be less than N $(m < N)$.

The security of AGS techniques is equivalent to the nonexistence of an adversary capable of existentially forging an AGS. The question is whether the adversary tries to forge an aggregate signature on a message of his choice. The adversary A is given a single private key. His goal is the existential forgery of an AGS. We allow the adversary to choose all private keys except the challenge private key. The adversary is also given access to a signing oracle on the challenge private key. His advantage AdvAGSA is defined to be his chance of success in the following game:

Setup. The aggregate forger A is provided with a private key d_1' generated at random.

Queries. A requests AGS with d_1' on the message of his choice.

Response. A outputs $t-1$ additional private keys $d_i', i = 2, t$ are included in A's forged AGS and an aggregate signature σ.

Definition: An aggregate forger A breaks a t-user AGS scheme in the aggregate chosen key model if the following conditions are met:

A runs in determinist time.

A makes $t - 1$ queries to the signing oracle.

A can guess (define) the challenge private key d_1' from σ, where $\sigma = \prod_{i=1}^{t} s_i$ and $s_i = h^{d_i'}$ with $i = 1, t$.

The proposed model is based on RSA, each node i has a public key and a private key (e_i, d_i') where $d_i' = d_i + r$ and $(m^{e_i})^{d_i} \mod N = m$; d_i and r are not known for nodes. Therefore, if RSA is secure, our scheme is more secure. The signing is meaningless if we say the private key has been revealed. Because in any encryption system, we will consider that the private key can only be accessed by its owner.

The adversary can easily obtain the private key if a technique is insecure. Currently, RSA is considered one of the strongest and most popular encryption techniques and the most widely used worldwide. No algorithm yet enables the private key "d" from the cipher, the signature, or the public key.

Since our scheme is based on RSA, we consider 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 $c = m^d \mod N = m^{d'} \mod N$ 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.

Transaction integrity relies on the collision and pre-image resistance of the Hash algorithm. SHA256 is based on the Merkle-Damgard arrangement, where the next block's hash value depends on the previous block's hash value, so it is expected to be collision-resistant due to the underlying compression function.

B. WHY MODIFIED RSA?

We use the same modulus N for all system participants in our setting. If TTP gives everyone a key pair (e_i, d_i) where $e_i \times d_i \equiv 1 \mod \phi(N)$, an adversary can get all private keys e_i and easily forge the signature and impersonate any other node in the system. The adversary gets two (e_1, d_1) and (e_2, d_2) where:

$$\begin{cases} e_1 \times d_1 \equiv 1 \mod \phi(N) \\ e_2 \times d_2 \equiv 1 \mod \phi(N) \end{cases}$$

Which gives:

$$\begin{cases} e_1 \times d_1 = 1 + \alpha \times \phi(N) \\ e_2 \times d_2 = 1 + \beta \times \phi(N) \end{cases}$$

we see that if e_1, e_2, d_1, and d_2 are known, then the adversary can extract $\phi(N)$; consequently, computing p and q; finally, computing all other privates keys d_i by using nodes' public keys e_i.

Thus, the TTP does not give node i the real d_i but the twin d_i' where $d_i' = d_i + r$ with keeping r secret. Now, the adversary can not get $\phi(N)$ regardless of its number of pair keys.

$$\begin{cases} (d_1' - r) \times e_1 = 1 + \alpha \times \phi(N) \\ (d_2' - r) \times e_2 = 1 + \beta \times \phi(N) \end{cases}$$

$$\begin{cases} d_1' \times e_1 = 1 + r \times e_1 + \alpha \times \phi(N) \\ d_2' \times e_2 = 1 + r \times e_2 + \beta \times \phi(N) \end{cases}$$

Even if e_1, e_2, d_1', and d_2' are known, we now notice that there is another secret variable r in the equations, which makes it impossible to calculate $\phi(N)$.

There are other known attacks on RSA signature, such as Existential forgery and Chosen message attacks. To mitigate these attacks, a message m is not signed directly as is ($\sigma = m^d$ where d is the private key), instead, m is encapsulated first using an encoding function like the Hash function, putting $h = Hash(m)$, then signing h as $\sigma = h^d$. The one-way function is provably secure in the random oracle model because the adversary could not compute the hash function by himself.

C. PREVENTING 51% AND ROGUE PUBLIC KEY ATTACK ATTACKS ON BLOCKCHAIN

In the basic blockchain implementation, a 51% attack requires the adversary to control 51% of computational power to alter backdated transactions and generate blocks with faulty transactions to achieve a double-spending attack. We can add another signature layer to increase the security level in the blockchain by avoiding the 51% attacks. To validate a new block, the system needs t' signatures; the miner diffuses the block and waits to get t' random signatures on this block to aggregate them. Therefore, computational power has no significance. The cooperation of t' malicious nodes is needed to attack the system. To validate its block by honest nodes, the malicious miner diffuses data and waits until receiving t' responses from its pool. Observing that an honest node receives responses randomly, it takes more than t' malicious nodes to perform a double-spending attack.

In basic practice blockchain, a rogue public key attack (RPKA) is possible because it is trivial to compute a new rogue public key using a subtraction operation on a given field G. Our technique is secure against RPKA without requiring every user to prove knowledge or possess the corresponding secret key. On the one hand, each pair of keys is authenticated by a trusted third party; on the other hand, there is a single d_i realizes $(m^{e_i})^{d_i} = m$.

VII. CONCLUSION

This article introduced a new aggregate signature scheme, AGS-MR, based on the RSA cryptosystem. We modified the basic RSA technique to improve security since AGS-MR uses the same modulus N for all network nodes. AGS-MR's architecture is designed to be applied in many settings requiring multiple signatures where the nodes do not need to prove knowledge or possess their secret key. Regardless of the number of nodes in the network, the final aggregate signature size in AGS-MR is equal to $O(k)$, where k is a security parameter.

In AGS-MR, all interactions between nodes are removed except in two rounds between the miner and signers, which gives $t + n$ exchanged messages instead of $n \times n$ in interactive multi-signature schemes. The specifications of AGS-MR are explained and compared to other schemes, including the current application of Bitcoin, in terms of communication and computation costs, where AGS-MR achieved $146KB$ vs. $216KB$ of its rival (Boneh AMSP) in terms of the size of AGS. Also, AGS-MR achieved 23.4 ms vs. 91.38 ms compared to the best rival (Cui et al.) regarding computation cost for 203 signatures. Concerning network traffic, the AGS-MR scheme significantly reduces the number of exchanged messages; it achieved $t + n$ vs. $t \times t$ messages, where t is the number of signers and n denotes the network size. In future works, we intend to find a more efficient method to compute the aggregate public key to be independent of the number of signers in terms of size.

REFERENCES

[1] G. Epiphaniou, P. Pillai, M. Bottarelli, H. Al-Khateeb, M. Hammoudesh, and C. Maple, "Electronic regulation of data sharing and processing using smart ledger technologies for supply-chain security," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1059–1073, Nov. 2020.
[2] S. Moffat, M. Hammoudeh, and R. Hegarty, "A survey on ciphertext-policy attribute-based encryption (CP-ABE) approaches to data security on mobile devices and its application to IoT," in *Proc. Int. Conf. Future Netw. Distrib. Syst.*, Jul. 2017, pp. 1–10.
[3] K. Chait, A. Laouid, L. Laouamer, and M. Kara, "A multi-key based lightweight additive homomorphic encryption scheme," in *Proc. Int. Conf. Artif. Intell. Cyber Secur. Syst. Privacy (AI-CSP)*, Nov. 2021, pp. 1–6.
[4] M. Kara, A. Laouid, A. Bounceur, M. Hammoudeh, and M. Alshaikh, "Perfect confidentiality through unconditionally secure homomorphic encryption using otp with a single pre-shared key," *J. Inf. Sci. Eng.*, vol. 39, no. 1, pp. 183–195, 2023.
[5] E. Makri, D. Rotaru, F. Vercauteren, and S. Wagh, "Efficient comparison for secure multi-party computation," in *Proc. Int. Conf. Financial Cryptography Data Secur.* Cham, Switzerland: Springer, 2021, pp. 249–270.
[6] M. Kara, A. Laouid, A. Bounceur, M. Hammoudeh, M. Alshaikh, and R. Kebache, "Semi-decentralized model for drone collaboration on secure measurement of positions," in *Proc. 5th Int. Conf. Future Netw. Distrib. Syst.*, Dec. 2021, pp. 64–69.
[7] D. W. Archer, D. Bogdanov, Y. Lindell, L. Kamm, K. Nielsen, J. I. Pagter, N. P. Smart, and R. N. Wright, "From keys to databases—Real-world applications of secure multi-party computation," *Comput. J.*, vol. 61, no. 12, pp. 1749–1771, 2018.
[8] M. Hammoudeh, B. Adebisi, D. Unal, and A. Laouid, "Bringing coordination languages back to the future using blockchain smart contracts," in *Proc. 5th Int. Conf. Future Netw. Distrib. Syst.*, Dec. 2021, pp. 299–304.
[9] A. Habib, A. Laouid, and M. Kara, "Secure consensus clock synchronization in wireless sensor networks," in *Proc. Int. Conf. Artif. Intell. Cyber Secur. Syst. Privacy (AI-CSP)*, Nov. 2021, pp. 1–6.
[10] Q. He, X. Xin, and Q. Yang, "Security analysis and improvement of a quantum multi-signature protocol," *Quantum Inf. Process.*, vol. 20, no. 1, pp. 1–21, Jan. 2021.
[11] H. Zhang, X. Zou, G. Xie, and Z. Li, "Blockchain multi-signature wallet system," in *Proc. CCF China Blockchain Conf.*, Wuxi, China: Springer, Dec. 2022, p. 31.
[12] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Found. Secure Comput.*, vol. 4, no. 11, pp. 169–180, 1978.
[13] K. Itakura and K. Nakamura, "A public-key cryptosystem suitable for digital multisignatures," *NEC Res. Develop.*, vol. 71, pp. 1–8, Jan. 1983.
[14] D. Boneh, "Aggregate and verifiability encrypted signature form bilinear maps," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 2003, pp. 416–432.

[15] K. Qiao, H. Tang, W. You, and Y. Zhao, "Blockchain privacy protection scheme based on aggregate signature," in *Proc. IEEE 4th Int. Conf. Cloud Comput. Big Data Anal. (ICCCBDA)*, Apr. 2019, pp. 492–497.
[16] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Cham, Switzerland: Springer, 2018, pp. 435–464.
[17] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, Oct. 2006, pp. 390–399.
[18] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple Schnorr multi-signatures with applications to bitcoin," *Des., Codes Cryptogr.*, vol. 87, no. 9, pp. 2139–2164, Sep. 2019.
[19] Y. Zhao, "Practical aggregate signature from general elliptic curves, and applications to blockchain," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Jul. 2019, pp. 529–538.
[20] M. Blanton and M. Jeong, "Improved signature schemes for secure multi-party computation with certified inputs," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2018, pp. 438–460.
[21] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2004, pp. 56–72.
[22] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 469–472, Jul. 1985.
[23] K. Lee and H. Kim, "Two-round multi-signatures from Okamoto signatures," *Mathematics*, vol. 11, no. 14, Jul. 2023, Art. no. 3223. [Online]. Available: https://www.mdpi.com/2227-7390/11/14/3223, doi: 10.3390/math11143223.
[24] T. Li, H. Wang, D. He, and J. Yu, "Permissioned blockchain-based anonymous and traceable aggregate signature scheme for industrial Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8387–8398, May 2021.
[25] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 5, pp. 840–852, Sep. 2018.
[26] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi, "Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices," *J. Cryptol.*, vol. 35, no. 2, pp. 1–56, Apr. 2022.
[27] V. Lyubashevsky, "Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures," in *Advances in Cryptology—ASIACRYPT*. Berlin, Germany: Springer, 2009, pp. 598–616.
[28] R. Choi and K. Kim, "Lattice-based multi-signature with linear homomorphism," in *Proc. Symp. Cryptogr. Inf. Secur. (SCIS)*, 2016, pp. 1–8.
[29] M. Bellare and W. Dai, "Chain reductions for multi-signatures and the HBMS scheme," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Singapore: Springer, Dec. 2021, pp. 650–678.
[30] Y. Gao and J. WU, "Efficient multi-party fair contract signing protocol based on blockchains," *J. Cryptologic Res.*, vol. 5, no. 5, pp. 556–567, 2018.
[31] Y. Zhao, "Aggregation of gamma-signatures and applications to bitcoin," *IACR Cryptol. ePrint Arch.*, 2018, Art. no. 414. [Online]. Available: https://ia.cr/2018/414
[32] J. Cui, J. Zhang, H. Zhong, R. Shi, and Y. Xu, "An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks," *Inf. Sci.*, vols. 451–452, pp. 1–15, Jul. 2018.
[33] H. Shu, P. Qi, Y. Huang, F. Chen, D. Xie, and L. Sun, "An efficient certificateless aggregate signature scheme for blockchain-based medical cyber physical systems," *Sensors*, vol. 20, no. 5, p. 1521, Mar. 2020.

KHALED CHAIT received the Eng. and M.Sc. degrees in computer engineering from the University of Bejaia, in 2008 and 2011, respectively. He is currently pursuing the Ph.D. degree with the University of El Oued. Until 2013, he was a Research Assistant with the Research Centre for Scientific and Technical Information (CERIST), Algeria, later with industry, as a Software Engineer. His research interests are distributed systems, artificial intelligence, security, and blockchain.

ABDELKADER LAOUID received the Eng. degree from Batna University, in 2008, the Magister degree from the University of Bejaia, in 2011, and the Ph.D. degree from the University of Bretagne Occidental, in 2017. He is a Full Professor and the Head of the Artificial Intelligence and its Applications Laboratory, University of El Oued. He is a researcher in the field of computer science, with a particular focus on cybersecurity, the IoT, distributed systems, and blockchain.

MOSTEFA KARA received the Eng. degree in computer science from the University of Biskra, Algeria, in 2005, and the master's degree in artificial intelligence and the Ph.D. degree in advanced information systems from the University of El Oued, Algeria, in 2019 and 2022, respectively. His research interests include computer security, cryptography, and blockchain.

MOHAMMAD HAMMOUDEH received the B.Sc. degree in computer communications from Arts Sciences and Technology University, in 2004, the M.Sc. degree in advanced distributed systems from the University of Leicester, in 2006, and the Ph.D. degree in computer science from the University of Wolverhampton, in 2008. He is the Saudi Aramco Chair Professor of cyber security with the King Fahd University of Petroleum and Minerals. His research interests include the applications of zero trust security to internet-connected critical national infrastructures, blockchains, and other complex highly decentralized systems.

OMAR ALDABBAS received the B.E. degree from Philadelphia University, in 2003, and the M.Sc. and Ph.D. degrees from De Montfort University, Leicester, U.K., in 2006 and 2008, respectively, all in computer engineering. Since 2016, he has been an Associate Professor and the Director of the Consultations, Studies, and Training Center, Al-Balqa' Applied University.

ABDULLAH T. AL-ESSA is a Cybersecurity Professional with Saudi Aramco Cybersecurity Operations. He has ten years of experience in various cybersecurity domains. He led multiple teams within the organization. He is currently the Head of Cryptography Solutions.