A Blockchain-Based Secure Online Crime Reporting and Management System

1.Pale Sushmitha
Department of Artificial intelligence And Data Science
Dhanalakshmi Srinivasan University
Tamil Nadu, India
Sushmithapale18@gmail.com

2.Pollampalli Navya
Department of Artificial Intelligence And Data Science
Dhanalakshmi Srinivasan University
Tamil Nadu, India
navyapollampalli@gmail.com

3.Ravuri Pushanjali
Department of Artificial Intelligence And Data Science
Dhanalakshmi Srinivasan University
Tamil Nadu, India
ravuripushpanjali5@gmail.com

Guide: Mrs. M Suguna
Assistant Professor
Department of Artificial Intelligence and Data Science
Dhanalakshmi Srinivasan University
Suguna15.9@gmail.com

1. Introduction

In 2008, Satoshi Nakamoto employed blockchain (BC) technology as the foundational infrastructure for the Bitcoin project [1]. While the concept of BC had been proposed earlier, predating the emergence of Bitcoin, its utilization extended to diverse applications such as filtering spam emails. Nevertheless, BC, particularly as the underlying technology of Bitcoin, has gained significant prominence in the past decade. BC technology constitutes a distributed database (DB) accessible to all network users. Comprising a sequence of blocks, it functions as a comprehensive ledger, recording transactions akin to a traditional public ledger [2,3]. The BC can be classified into three principal categories: public BC (open to everyone for participation, transactions, and access to stored data), consortium BC (tailored for multiple companies seeking collaborative ventures through BC technology), and private BC (exclusive to a specific company's personnel, barring external participation) [4].

The mining process serves the purpose of generating and submitting the succeeding block within certain BC systems. In the Bitcoin blockchain, individuals known as miners, who are active participants in the network, play a role in the submission of transactions onto the BC. In alternative BCs, the reward is conferred upon the initial miner or validator who successfully discovers or verifies the next valid block. Various methodologies exist for determining the deserving miner or validator for reward allocation, including proof of work (PoW), proof of stake (PoS), proof of space (PoSpace), proof of importance (PoI), measure of trust (MoT), and practical Byzantine fault tolerance (PBFT) [1,5]. It is pertinent to note that not all BCs necessitate the mining process for solving a computational puzzle. For instance, PBFT eschews this mechanism and instead relies on a consensus achieved through message passing.

Bitcoin employs the PoW mechanism to facilitate the submission of transactions on the BC. In the Bitcoin blockchain, each miner endeavors to solve the PoW puzzle, a computational challenge aimed at creating the next valid block for transaction submission on the BC, and in turn, securing rewards. Consequently, bitcoins are allocated as rewards to the initial miner who successfully identifies the valid block [1]. The acquisition of bitcoins as a reward motivates all miners to engage in solving the PoW puzzle. However, only one miner emerges victorious, resulting in the expenditure of energy by unsuccessful participants. An additional

illustration of the PoW puzzle is evident in the Ethereum project, introduced in 2014 [6]. Notably, Ethereum has transitioned to a different consensus mechanism known as proof of stake (PoS) with Ethereum 2.0 (ETH 2.0) [7].

Following the introduction of BC technology in 2008, a transformative wave swept through the industry. The inherent characteristics of decentralization, transparency, open-source architecture, autonomy, immutability, and anonymity [4] stimulated developers and researchers to explore its applications. Capitalizing on these features, cryptographers extended the use of BC technology, leading to the development of numerous cryptocurrencies. This technology was then integrated with other cryptographic primitives, enabling its application in diverse domains such as the industrial internet of things (IIoT) and smart grid networks (SGNs) [8,9], e-government initiatives [10], e-voting systems [11], and healthcare applications [12,13].

Consortium or Private BC-based schemes are typically initiated and overseen by a single entity, often the BC owner. This entity takes the initiative to initialize the BC and deploy associated smart contracts. Despite the inherent transparency of BC technology, the majority of proposed schemes tend to consider transactions submitted by the central authority as automatically valid on the BC. A case in point is the *Reportcoin* project [14], where the verifier, after receiving reports, exercises discretion in verifying and subsequently submitting them to the blockchain. The verifier's decision to submit a received report on the BC is subjective, introducing the possibility of policy violations and non-submission. This characterization of the verifier as an untrusted party, deviating from protocol phases, contradicts the distributed implementation ethos intrinsic to BC-based systems. A noteworthy observation is the potential to replace certain entities with verifiers in these schemes. This substitution introduces the prospect of multiple parties managing the BC-based network, with decision-making authority contingent on meeting a specified threshold. Even in scenarios where some parties may be deemed untrusted, the BC-based network can still operate effectively. This conceptual shift aims to foster a more distributed governance model, where the collective decisions of multiple entities contribute to the integrity and functionality of the BC-based network.

Historically, democracy has consistently been a desired governance principle among community members, reflecting the will of the majority [15]. Despite this, certain governments operate without democratic principles, where decision-making authority is consolidated in a single individual [16]. Such autocratic structures are generally unwelcome in most communities, as individuals seek to participate in decisions that affect them. This sentiment extends to computer networks, where users aspire to actively engage in monitoring and influencing decisions within their network. However, the presence of a network administrator often serves as the entity responsible for implementing network policies. Similarly, in several blockchain (BC)-based systems and protocols [8–12], there exists an assumption of a fully trusted entity. As highlighted previously, this approach is not tenable for users of public BCs, as it contradicts the foundational principles of decentralization and user empowerment. In public BCs, the expectation is for a more distributed and inclusive governance model, ensuring that users maintain agency over their network and are not subject to centralized decision-making authorities.

In recent times, there has been a growing consideration for anonymous reporting systems leveraging BC technology. In these systems, a CA refrains from submitting a report on the BC when deemed a potential malicious actor. This cautious approach is taken because a report submitted by a potentially compromised CA could be unfavorable and influenced by the reported incident, thereby diminishing network reliability. Nevertheless, we contend that there is a perceived need for the development of a BC-based reporting protocol that does not rely on any individual party, such as the CA. The absence of such a protocol is evident and represents a gap in the current landscape of BC-enabled reporting systems.

1.1. Contribution and methodology

In this paper, we introduce a blockchain-based system termed "Anonymous Reporting System with No Central Authority (*ARSnCA*)." The system encompasses both an architectural framework and a comprehensive protocol. Within the *ARSnCA* architecture, the process of report submission diverges from reliance on a singular entity, such as the central authority (CA). Instead, each submitted report undergoes reconstruction and approval by a group of members. Crucially, the approved report can then be submitted on the blockchain as a valid report, with the condition that at least one member of the group completes the submission. To devise the *ARSnCA* protocol in a manner that fulfills the specified properties, we employ a range of techniques and methodologies outlined below.

- To remove CA, we use a concept called *Virtual Blockchain* (VBC) that is defined as a private blockchain embedded in a public blockchain. The CA is replaced by VBC in BC-based networks to remove CA's individual decisions (by assigning CA's duties to the group of members).[1]
- To assign network decisions to VBC members, we apply a combination of secret-sharing protocol (SSP) and asymmetric encryption.
- To protect BC members' (who send reports) and VBC members' (who reconstruct and submit reports) privacy, we apply ring signature as a cryptographic tool that provides user anonymity and preserves their privacy.

The *ARSnCA* protocol's comparison and evaluation show that it is the fastest protocol among all if 5 users are present in used ring signatures (generally, the number of users □ between 5 to 10 is used as a practical number for ring signatures [14]). Moreover, by the assumption that the number of present users in used ring signatures is 10, the *ARSnCA* protocol is yet 62% faster than a blockchain-based e-voting protocol, and it is 92% faster than a blockchain-based anonymous reputation protocol.

1.2. Organization

The rest of paper is organized as follows: In Section 2, we describe some network models in the category of the distributed management systems (DMS). We then explain the paper preliminaries in Section 3. In Section 4, we suggest the *AREnCA* architecture. In Section 5, we present the *AREnCA* protocol and analyze it. Finally, we compare the *ARSnCA* protocol with recently proposed protocol in Section 6.

2. Related works

Initially, it is imperative to underscore that this section discusses a limited body of existing works. This scarcity is attributed to the novelty of the concept of developing a reporting protocol based on blockchain technology. As of now, there are only a few studies that specifically concentrate on blockchain-based reporting protocols.

In 2019, Liu et al. proposed a BC-based anonymous reputation system for IIoT using PoS [8]. The Liu et al.'s scheme preserves the reviewer anonymity and accountability while designing a versatile anonymous rating token. There are three entities in their presented network model: identity manager (IDM), retailer, and consumers. According to their assumption, IDM is a government agency that can issue and manage the identities and credentials of consumers and retailers, and consumers can give reputations to retailers anonymously.

Tao et al. proposed an anonymous authentication protocol (AAP) for SGNs [9]. Their assumed SGN network model consists of the RA

[1] To be the defined concept called "VBC" and "blockchain virtualization" that needs to be described: Virtualization is a service divides each thing into some separate parts so that each part works independently and without help from others. However, VBC is replaced by CA. VBC members are embedded in the same blockchain, and they do not work separately from the public blockchain.

(registration authority), ESs (edge server), EUs (end-user), and the BC is assumed as a distributed ledger. The assumption in this scheme is that the RA is a services provider and trusted by all EUs. One of the security requirements is mentioned in the [9] is "no online RA" since they want to minimize communication overhead and resist against a single point of failure.

In 2019, Tang et al. presented another BC-based protocol for the health records [12]. The electronic health records (EHR) network model includes three levels: Level−0 is the server level, level−1 contains other organizations, and level−2's entities correspond to the employees of the level−1. It is clear, in the presented network architecture, access control is assumed for each level, and the server has more authority than others, and it (or privileged entities in Level−0) can decide individually against network users' tastes if it is assumed as an untrusted party.

The other presented scheme in 2019 is an anonymous reporting system based on BC named *Reportcoin* [14] by Zou et al.. To protect users' identities, authors applied a ring signature. In the *Reportcoin*, the users can cooperate to create a ring signature on the report and send it to the verifier for approval. As we know, the ring signature is primitive, so that its computational cost and communication overhead have a linear relation to the number of users. Another reporting protocol that was proposed in the same year is a blockchain-based anonymous reporting with anonymous rewarding named BB2AR [17]. The BB2AR's authors have acted similar to *Reportcoin* [14]. They applied a ring signature for user anonymity. The method they used to give anonymous rewarding is based on [18] that presented the concept of blockchain-based anonymous rewarding for the first time in the same year.

In 2020, Lin et al. presented a blockchain-based protocol that provided conditional privacy-preserving in vehicular network [19]. The BLS [20] signature and a VANET-based anonymous authentication protocol [21,22] are used in the BCPPA protocol for providing authentication and anonymity for vehicles, respectively.

In 2021 and as our previous work, we proposed the first quantum-secure reporting protocol (*QS-RP*). To provide security against quantum computers, we reduced the *QS-RP*'s security to the two quantum-secure cryptographic primitives, including hash function and multivariate public-key cryptography. The *QS-RP* supported some attractive security features such as report confidentiality before verification, whistleblower anonymity, and public verification for the submitted report.

Discussion: Several common characteristics are evident across the discussed schemes. Firstly, all schemes exhibit a distributed nature (□). Secondly, transparency is a shared attribute, with all schemes ensuring openness in their operations (□□). Additionally, every member in these schemes possesses access to the distributed DB developed on the public blockchain (□□□). Despite these commonalities, there exists a notable entity, often referred to as the CA or by alternative names such as RA (Registration Authority), IDM (Identity Management), Authority, or Certificate Authority. In our perspective, this entity is considered potentially untrusted. Table 1 outlines key information regarding this central authority in the mentioned schemes. The table is structured with two initial columns listing the entities' names in the schemes. The third and fourth columns provide additional details about the central authority, encompassing its description, duties, and capabilities. To illustrate, in Tao et al.'s scheme [9], the RA is endowed with the capability to individually update or revoke users' keys.

3. Preliminaries

The preliminaries of this paper will be described in this section, and the list of used notations is shown in Table 2.

3.1. Ring signature

A ring signature is a well-known cryptographic primitive that provides user authentication and user anonymity in a group of network users [24], and it is defined through four probabilistic polynomial-time (PPT) algorithms as follows.

- (□□□□□□) ←*Setup*(□): The *Setup* algorithm takes the security parameters □ and returns the list of system public parameters □□□□□□ and the system secret key □.
- (□□□$_{□□}$, □□□$_{□□}$) ←*KeyGen*(□□□□□□, □): On the list of system public parameters □□□□□□ and the system secret key □, the pair of public-private keys □□□$_{□□□}$ − □□□$_{□□}$ for each user are returned by the *KeyGen* algorithm.
- (□) ←*RSign*(□□□, □□□$_{□□}$, □□□□$_{□□}$): On message □□□, the list of public keys □□□□$_{□□□}$, and private key □□□$_{□□□}$ the *RSign* algorithm returns the ring signature □.
- ({1, 0}) ←*RVerif*(□, □□□□$_{□□}$, □□□): The ring signature □, the list of public keys □□□□$_{□□□}$, and the message □□□ are given to the *Verif* algorithm, and it verify the validity of the ring signature.

According to the definition of the typical ring signature, □) the list of public keys □□□□$_{□□□}$ is used in the verification process, and it shows that the user who created the ring signature □ is valid (the user is authenti- cated), and □□) the user's public key is not used directly in the verification process, and the user who created the ring signature □ will be anonymous among the group of users. Therefore, the anonymity for a typical ring signature scheme is defined as below:

Definition (anonymity) *A typical ring signature scheme provides anonymity, and no one can find the user who creates the ring signature. Moreover, no one can find a link between a user who created two ring signatures twice. The advantage of a PPT* A *who wants to violate user anonymity and user unlinkability is written as* □□□ $^{□□□□□□□□□}_{A}$ = $^{□□□□□□□□□}$□□□$_{A}$ + $^{□□□□□□□□□□□□□}$□□□$_{A}$ < □.

The two written advantages are calculated as below:

- □□□ $^{□□□□□□□□□}_{A}$ = □ □[A(□□□□□□, □, □□□□$_{□□}$, □□□) = □□□$_{□□}$] < □ that means a PPT A has a negligible advantage to find the public key of the user, who create the ring signature, on having □□□□□□, □, □□□□$_{□□□}$, □□□.
- □□□ $^{□□□□□□□□□□□□□}_{A}$ = □ □[A(□$_1$, □$_2$, □□□$_{□□}$) = □$_1$ ↔ □□□$_{□□}$ |□ ∈ {1, 2}] − $\frac{1}{2}$ < □ that means for given two different ring signatures □$_1$ and □$_2$ and two different public keys □□□$_{□□_1}$ and □□□$_{□□_1}$ to a PPT A, it has a negligible advantage to find □) a link between two users and their two created ring signatures, or □□) a user who created two ring signatures at two different times.

In this study, a ring signature is employed as a cryptographic primitive to afford anonymity to the reporter or whistleblower. Furthermore, it is essential to recognize that signature schemes, including the ring signature scheme used in this work, are acknowledged as robust authentication methods. Consequently, each reporter or whistleblower submitting a report for verification undergoes anonymous authentication, establishing them as a valid user. If a submitted report does not pass this authentication, it is not processed further. This ensures that only authenticated and valid users are authorized to contribute to the reporting system.

3.2. Combination of two cryptographic primitives

In this section, we first review the two cryptographic primitives and then define a combination of them.

- *Trapdoor function:* One of the cryptographic problems that was solved in 1976 by Diffie and Hellman is "finding a function that has a trapdoor" [25]. The widespread application of trapdoor one-way functions is to apply it in asymmetric encryption [26,27].
- *Secret sharing:* Secret sharing (SS) means that a person can divide a □□□□□□ into □ shares and distributes them to □ parties. To reconstruct the original □□□□□□, all □ parties have to be present and cooper- ate [28,29]. Another kind of SS is threshold secret sharing protocol

Table 1
The Summary of aforementioned DMS schemes.

Description ⇒ Scheme ⇓	Users type	Central entity name	Who is it? What is its description?	What are its duties? What are its abilities?
Liu [8]	Consumers, and Retailers	Identity manager (IDM)	1. Is a government agency. 2. To be fully trusted.	1. charge of issuing and managing identities and credentials of consumers and retailers. 2. Responsible for the administration of the citizens.
Tao [9]	End users, and Edge servers	Registration authority (RA)	1. Is the electricity service provider. 2. Is trusted by all participants in the smart grid system.	1. Tasked to distributed key materials for all participants. 2. Able to utilize the blockchain to record participants' key materials using smart contract for identity validation, key update and revocation.
Tang [12]	Hospitals and Doctors, Researchers, Insurance agents	Server (S)	1. Is in charge of the generation of system parameters.	1. Chooses public parameters for all users.
Zou [14]	Mobile terminal users	Verifier (F)	1. Is a mobile user. 2. Is a verifier.	1. Receives announcement. 2. Can verify announcement. 3. Creates the blocks.
Wang [17]	Regular users/ Whistleblowers	Authority	1. Is a central authority. 2. Is a trusted party. 3. Verifies the reports and pay for them.	1. Punishes the criminals. 2. Encourages the people to expose their crime evidence. 3. Reward the whistleblower if he submits the valuable materials.
Lin [19]	Vehicles	Certificate authority (CA)	1. Is a trusted entity with enough resources. 2. Is responsible for managing certificates of vehicles.	1. Signs certificates. 2. Issues keys. 3. It can obtain the real identity of the vehicles.
Banaeian Far [23]	Regular users/ Whistleblowers	Central authority (CA)	1. It is a fully trusted party who initializes the system.	1. It assigns auditors' private keys. 2. It verifies the report proofs and submits them on the blockchain.

Table 2
The List of Notations.

Notation	Description	Notation	Description
A	The adversary	□	The generator of □(□□)
□□□	The array consists of □ − 1 random number □	□	The generator of □
□	The random challenge (used in □□1)	□□□□	The □□□'s private key
□′	The random challenge (used in □□2)	□□□□	The □□□'s private key
□□□□ (.)	Decryption using □□□'s private key	□□□□	The □□□'s public key
□□□□ (.)	Encryption using □□□'s public key	□□□□	The □□□'s public key
□(□□)	The Elliptic curve	□	Prime order of □ on elliptic curve □(□□)
□	The cyclic additive group	□	The length of □□□□□□□
h(.)	Secure one-way hash function	□	The random number in RS1
□	The security parameter	□′	The random number in RS2
□	The list of VBC members	□□□□□	The random number in □ □□□ function
□□□	The identity of □th VBC member	□	The secret key of CA
LEN	The length of VBC member	□□□□□□	The original secret
□□□	The length of BC member	□□□□□□_□□□	The one-time secret key using in □ □□□
□□□	The list of public keys	□	The decision threshold (%)
□□□□□□□	The list of VBC members' public keys	□□□□□□□□	The current time
□□□□□□□□	The list of selected BC members' public keys	‖	The concatenate operation
□	The point on the □(□□) (used in □□1)	□□□	The identity of □th BC member
□′	The point on the □(□□) (used in □□2)	□□□□(□, □)	A secure signature algorithm where □ is private key and □ is message
□□□	The all concatenated RRPs	□ □□□□(□, □)	The verification algorithm for □□□□(□, □) where □ is public key
□□□	The message	□	A random number used for update/regenerate private key
□	The length of □		

(TSSP), which means if some parties are absent, other existing parties can reconstruct the □□□□□□ [30,31]. The TSSP schemes should have *security* in such a way that no information about the □□□□□□ is obtained if the number of present participants is lower than the determined threshold [32,33].

Regarding the two above concepts, in the following, we define the two functions/algorithms of □ □□□ and □ □ □□□ (details of two mentioned functions are written in Section A.1.1, and they are analyzed in Section A.1.3). As a brief and based on notations described in Table 2, we divide a □□□□□□ into □ parts □□□□□□□ where 2 < □ ≤ □ and sends a part to all VBC members' public keys/addresses □□□□□□□ (for details see Section A.1). This section shows that all VBC members have to cooperate, assuming □ = 100 %.

In the following, the two functions of □ □□□ and □ □ □□□ are defined.

- (□□□□, □□□□, □□□□□□□□) ← □ □□□ (□, □□□, □□□□□□): The threshold □, message/report □□□, and the list of public keys/addresses □□□□□□□ (1 < □ ≤ □ and □□□□□□□ = □) are given to the □ □□□ function, and it returns □ 3-tuples {□□□□, □□□□, □□□□□□□□} where the □th tuple is related to the □th index of □□□□□□□ (this function is the combination of two phases of the presented method in Section A.1.1; □) *preparation phase* and □□) *transaction phase*. To see numerical examples of other values for the threshold □ refer to Section A.1.2).
- {□□□□□□□, ⊥} ← □ □ □□□ (□□□□, □□□□, □□□□□□□□): We define this function according to the *verification phase* of the presented method in Section A.1.1. In this function, three parameters □□□□, □□□□, and □□□□□□□□ are taken as inputs and returns □□□□□□□ if □□□□□□□ is obtained successfully. Else, it returns ⊥.

We use the two above defined algorithms in the *ARSnCA* protocol. To see the other modes of the defined functions and numerical examples with other □s, you can see Section A.1.2. Note that, if □ determined in

its minimum value □ = $\frac{100}{□}$% means that all VBC members □□□ s have an equal authority to CA.

3.3. Problem statement

In this section, a challenge and its solution are defined. Then, requirements and goals related to the solution are described.

3.3.1. The problem

This paper identifies a potential challenge prevalent in certain blockchain-based protocols, specifically focusing on the assumption that the CA is an untrusted party. Under this assumption, the CA holds the capability to independently control network users and private correspondences based on its discretion. Such a scenario introduces a considerable risk, as it has the potential to diminish network reliability. This challenge is particularly noteworthy in reporting systems, where there exists a possibility that the CA or other privileged insiders, influenced by the content of the submitted report, may attempt to disrupt the report submission process by engaging in activities such as altering, deleting, or corrupting the submitted reports. Addressing this challenge becomes imperative to ensure the integrity and trustworthiness of the reporting system.

3.3.2. The solution

This study proposes a novel approach by substituting more than one party for the CA and other privileged insiders within BC-based networks. It is essential to note that the selected parties must be implemented on the same BC infrastructure as the CA and the overall system. To facilitate this substitution, we introduce the replacement process as an embedded permissioned blockchain within the same permission-less blockchain, referred to as the "virtual blockchain" (VBC). The VBC is designed with the belief that it can effectively fulfill the required features while concurrently enhancing network and system fault tolerance and reliability. This innovative configuration aims to contribute to a more resilient and reliable BC-based network architecture.

3.3.3. Security requirements, threat model, and design goals

In this section, the threat model and security requirements will be described. We now define the goals that the suggested solution (*ARSnCA* architecture and protocol) should achieve. To analyze the *ARSnCA* protocol, the threat model [34,35] and adversary's (A) abilities will now listed in the following:

- The A, as an unregistered user, has the ability to eavesdrop, corrupt, and send every report to the VBC members for verification.
- The A can be assumed as the system owner who want to violate users' privacy.
- The A, as a registered user, can be present among all BC members.
- The A can be a candidate and can be selected as a VBC member and commit fraud.
- The A has the ability to control 100 − □ % of VBC members (it has access to some private keys of 100 − □ % valid VBC members).
- The A can be present on the public channel, and it can eavesdrop, corrupt, and intercept parameters sent on the public channel.
- The A has access to the stored data and transaction on the blockchain.

To measure A's winning probability, three functions are defined: experiment function (□□□), the success function (□□□), and the proba- bility function.(□ □) In theexperiment function, A performs the process to gain his required information. The success function, specifies how successful is A in gaining important data. Finally, the possibility func- tion is A's probability of success in the recovery of secret values. If the probability of success is negligible (□), the protocol is secure against assumed A as □□□□$^{□□□□□}$ = □ □[$_{A}^{□□□\ □□□□-□\ □□□□□}$] ≤ □.

We now define a formal model used for the security analysis of the *ARSnCA* protocol. First, we describe k*eveal*, that empowers A to guess and find unknown parameters. k*eceive* and S*end* are the queries that mean A can send/receive data to server/users. On the other hand, it can block user flows and S*end*s its dummy flow to server. C*urrupt* is the last query we define. It means A can currupt the recorded data on the blockchain. All in all, A is able to send queries including k*eveal*, k*eceive* and S*end* data or can C*urrupt* the parameters stored in the user memory or server database.

In the following, we present some security requirements [36–38] and define six goals that should be achieved in the suggested solution.

- *Indistinguishability:* Indistinguishability is one of the security features that security protocols require it. In this feature, we expect the attacker has no ability to link between two plain messages and two encrypted messages with a probability of higher than $\frac{1}{2}$.
- *User untraceability:* This feature means if a user logs in to a server several times or logs in to several servers, no one can find □) a link between sent log-in requests and the user, or □□) a link between the previous user and the current user.

  Goal 1. (Untraceability) *The ARSnCA protocol provides untraceability, and no PPT* A *can find* □) *a link between sent log-in requests and the user, or* □□) *a link between the previous user and the current user with a probability higher than* $\frac{1}{2}$.
- *User anonymity:* User anonymity includes two security features. User untraceability (we explained in the previous item) and user pseudonymity that means the real user's identity cannot be obtained by eavesdropping on the corresponded messages through public channels, access DB, or other hardware.

  Goal 2. (User pseudonymity) *The ARSnCA protocol provides user pseudonymity, and on having sensitive information, no PPT* A *can find bits of knowledge about the user's real identity.*
- *Replay attack:* To mislead the receiver, A tries to send each sent message again. The A uses this type of attack when it wants to gain some responses instead of the real user, or it wants to access something or somewhere as a valid user.

  Goal 3. (Resistance against replay attack) *The ARSnCA protocol provides resistance against replay attack, and repetitive messages are detected.*
- *Malicious insider attack:* In this attack, it is assumed A can be present among privileged insiders (or some of the privileged insiders are assumed malicious parties) and tries to make disorders in the executing processes or violates the determined policies.

  Goal 4. (Resistance against malicious insider attack) *The ARSnCA protocol provides security against insider attack, and privileged insiders cannot implement disorders in or violate the policies.*
- *DoS and DDoS attacks:* When the requests (or loads) sent to a server are high, the server cannot provide its services. The Denial of Service (DoS) attack means that A sends many requests to a specific server to make it unable to provide the services. The Distributed Denial of Service (DDoS) attack means that A sends many requests to a specific server from several locations (dealing with the DDoS attack is more challenging than dealing with a DoS attack).

  Goal 5. (Resistance against DoS and DDoS attacks) *The ARSnCA protocol is resisted against DoS and DDoS attacks, and no PPT* A *has the ability to make the ARSnCA protocol unable-to-service.*
- *Transparency:* Transparency means the database is accessible to all, and everyone can have a copy.

  Goal 6. (Transparency) *The ARSnCA protocol provides transparency.*

4. The *ARSnCA* architecture

In this section, we describe the "blockchain-based anonymous reporting system with no central authority (*ARSnCA*)" Architecture. At first, we define the existing components and entities in the *ARSnCA* Architecture. We then define the *ARSnCA* system model. The *ARSnCA* system model is shown in Figs. 1 and 2.

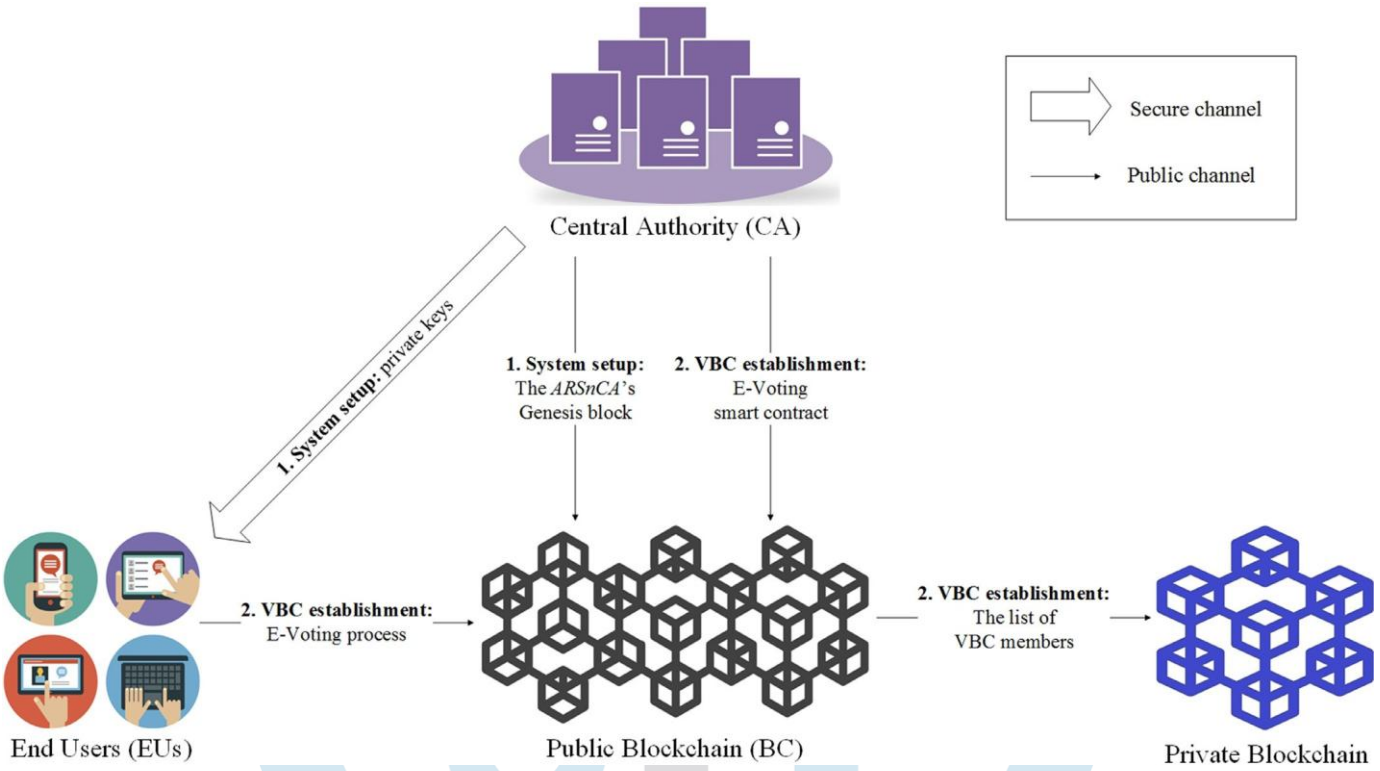

Fig. 1. The First Generation of the *ARSnCA*.

4.1. Entities and components

In the following and according to Figs. 1 and 2, the components and entities that exist in the *ARSnCA* system model will now be defined.

- *Blockchain (BC):* The BC is assumed as a public/ permissionless and distributed database that is responsible for queries.
- *Virtual blockchain (VBC):* The VBC is assumed as a private/ permissioned database that is responsible for the authorized users' queries (VBC members). The list of authorized users is updated periodically through an on-chain e-voting (or other fair ways) in each determined time period.
- *Central authority (CA):* Someone who initializes the system and is assumed as the system CA until forming the first version of virtual blockchain (VBC). After the VBC forming, CA has no authority, and all its duties are assigned to VBC members (the generated parameters by CA gets no advantage to it after VBC generating).
- *End users (BC members):* General users who can join the BC and have access the stored data. They also can join e-voting as candidates or voters, and they can be selected as VBC members or vote for their candidate.
- *VBC members:* Members of VBC who are selected via a fair e-voting for a specific time. They want to join VBC since they want to have a share in their network decisions (it is their motivation for join to VBC). They are replaced by CA and manage the public blockchain.

4.2. System model and data flows

We describe the system model of the *ARSnCA* protocol. The *ARSnCA* protocol has eight phases, including System setup, VBC establishment (these two phases are shown in Fig. 1), Find neighbors, Ring signature−1, Execute TSSP, Reconstruction and verification−1 on the VBC, Ring signature−2 and Submit on the BC, and verification−2 (these six phases are shown in Fig. 2). We describe the eight mentioned phases below.

1. *System setup:* The CA executes this phase and uses the security parameter □ to generate the system's public parameters □□□□□□. It keeps its private key □ secure. In this phase, the system genesis block that includes □□□□□□ and other needed parameters is submitted on the public BC by CA (the mentioned genesis block is referred to the first block related to the *ARSnCA*, not the genesis block of the main blockchain). Parts of BC members' private keys are sent to them through a secure channel. BC members have to keep their private key □□□$_{□□□}$ secure. Based on the part of the private key, each user selects a random number □ and updates its private key (to keep its secret key secure against malicious insiders).
2. *VBC establishment:* It is assumed that a secure BC-based e-voting or another fair way to select VBC members (this mechanism is out of this paper's scope but should be considered in the applicable smart contract) is executed in this phase, and all VBC members □$_{□□□}$s are chosen in this phase by BC members who want to vote for their candidates to establish the first generation of VBC and determine the threshold □. The VBC parameters are then published on the BC. After this phase, CA has no authority, and all tasks are assigned to VBC members (for details, refer to Tables 7 and 8, which illustrate the message sender should exist in the VBC member list □□□, not be the owner). It is clear that after this phase, VBC members are still BC members, but BC members are not necessarily a member of the VBC (Fig. 2 shows the system model after executing this phase).
3. *Find neighbors:* To create the ring signature, and for protecting the user's identity □$_{□□□}$, the user □$_{□□□}$ has to collect some witnesses (the list of its neighbors' public keys □□□□$_{□□□}$). Therefore, according to the three types of packets RQP, RRP, and RAP, defined and shown in Table 3, it tries to collect witnesses. It broadcasts the report (RQP packets) via a low-power broadcast system and waits for the RRP packets as witnesses, including public keys. Each user gets help from its neighbors to collect RRP packets for creating the ring signature using its private key and collected neighbors' public keys.
4. *Ring signature−1:* To submit the report □□□ on the BC, the user □$_{□□}$ has to send its report, using a ring signature, to the VBC members'

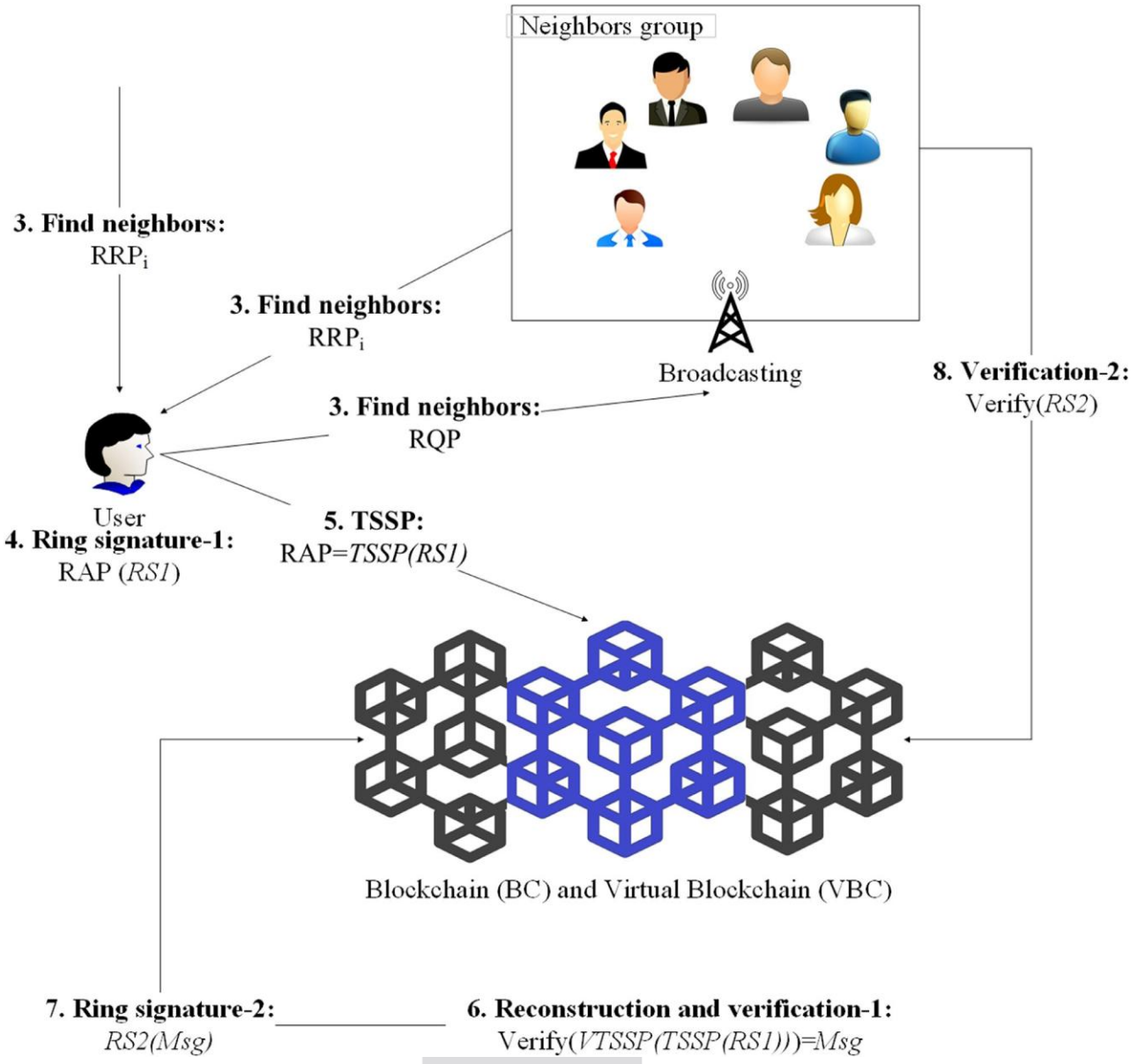

Fig. 2. The Second/Next Generation of the *ARSnCA*.

Table 3
Summary of three packet types.

Packet name	Generator	Transmission type	Receiver(s)	Reply type
RQP	□□□	Broadcast	BC members	Send back to □□□
RRP	BC members	Send to □□	□□□	No reply - Send to VBC members
RAP	□□	Send to □□□□□	□□□□□	No reply - Submit on the BC

public keys/addresses through a public channel. The □□□□ creates the ring signature (□□1) over the report □□□ using its private key □□□□□ and the collected neighbors' public keys □□□□□□□.

5. *TSSP:* To send the created □□1, the user □□□ executes the □ □□□ function/algorithm to get the report tuples. Then, the generated □ 3-tuples {□□□□, □□□□, □□□□□□□□} are sent to all VBC members publicly □□□□□□□ through a public channel.
6. *Reconstruction and verification−1 on VBC:* Upon receiving report tuples on VBC members' public keys/ addresses, they obtain their related 3-tuple {□□□□, □□□□, □□□□□□□□} from their public addresses □□□. The VBC members then cooperate to reconstruct □□1 with executing the □ □ □□□□ function, and verify the reconstructed □□1 that consists the report □□□ using □□□□□□□.
7. *Ring signature−2 and submit on BC:* After the reconstruction process of □□1, one (or more) of VBC members creates another ring signature (□□2) using VBC members' public keys □□□□□□ and its private key □□□□□ on the verified report. It then submits □□2 on the BC to be accessible for all.
8. *Verification−2:* All □□□ (BC members) can find and verify the submitted □□2 on the BC (it means they have access to the verified report). It should be noted that invalid report sent by malicious insider/outsider is not verified.

Table 4
System setup - 1.

Algorithm 1: BC - Initialization

```
contract BC {
address owner;
// We define the structure of components on the BC.
struct BC {
uint UIDi;
uint puUIDi;
}
BC[] public PBC;
constructor PBC() {
owner = msg.sender;
// First time
len = 0;
return 1;
}
}
```

Table 5
System setup - 2.

Algorithm 2: BC - User registration

```
function Ureg (UIDi, puUIDi) {
// We assume that there is no user and BC owner executes this algorithm.
if owner ≠ msg.sender then
return 0;
else {
len+;
BC[len].UIDi = UIDi;
BC[len].puUIDi = puUIDi;
return 1;
}
}
```

5. The *ARSnCA* protocol

In this section, we present the details of the *ARSnCA* protocol and analyze it.

5.1. The protocol

We describe the *ARSnCA* protocol in details in the following. Additionally, we write related pseudocodes in Tables 4 to 8.

5.1.1. System setup phase

The CA executes the *system setup* phase. This phase has two steps, including □) publishing system public parameters on the public BC and □□) user registration; we describe them below.

1. *Publishing system public parameters:* The CA uses the security parameter □ and generates system's public parameters □□□□□□. It then generates □□□□□□ = {□, □, □ = □□, □, □, $h(.)$, □(□), □, □(.), □ □□□, □ □ □□□} as a set of the *ARSnCA*'s public parameters (a system genesis block) and keeps its private key □ secure. The system genesis block that includes □□□□□□ is submitted on the BC by CA. We show the description of the mentioned parameters in Table 2. The pseudocode of this phase is shown in Table 4 (Algorithm 1).
2. *User registration:* In this step, the *ARSnCA* system has no members. To register users, CA sets the pair of public-private keys according to the proposed registration phase algorithm in [9] (see Section A.2). Then, users are added to the BC as shown in Table 5 (Algorithm 2). Users who are registered in this phase can vote for their candidate to establish the first generation of the VBC. After the VBC establishment, the updated users (who are updated by VBC members) can vote to their candidate to develop the second/next generation of the VBC after spending the specific time.

5.1.2. VBC Establishment phase

At first, we note that after this phase, all decisions are made by VBC members (e.g., updating BC members, adding a new member, or revoking current members), and all submitted transactions are assumed valid if they are verified using VBC members' public key. In this phase, a secure e-voting protocol is executed by BC members to select VBC members (it was shown in Fig. 1). Then, the list of selected VBC members □□□□□□□ is submitted on the BC. According to Fig. 2, the first generation of the VBC is established by CA (□□□ = 0), and current VBC members will generate other generations. We show this process in Table 6 (Algorithm 3).

Now, the VBC is established, and its members can update BC members as shown in Tables 7 and 8 (Algorithms 4 and 5, respectively). However, we note that each VBC member can execute this algorithm after receiving □□1, □, and the minimum length of □ that are determined in this phase (e.g., $3 \leq \square \leq 10$).

Remark. *How is it shown that the owner has no authority after this phase?* As the response, in Table 7 (Algorithm 4) and Table 8 (Algorithm 5), it was written that "if LIDi ≠ msg.sender then return 0;", which means only VBC members are assumed valid to send messages, and the owner has no authority after this phase.

5.1.3. Find neighbors phase

The user □□□□ has to have some witnesses or some users who support it if it wants to submit the report □□□ on the public BC. To achieve this goal, the user □□□□ broadcasts RQP=□□□□(□□□□□, □□□) and waits for the responses (RRP). After a specified time, □□□ receives some RRP= □□□□(□□□□□, □□□) where □ ≠ □ includes its neighbors' signature on it to sent the report □□□.

Now, the user □□□□ has some RRPs as its neighbors' witnesses and the list of its neighbors' public keys □□□□□□□ to create □□1 (ring signature is used for protecting □□□□'s privacy). The user □□□□ is a valid user to create □□1 if it has been received at least □ RRP, else it cannot create a valid □□1. Therefore, it has to collect more RRPs.

5.1.4. Ring signature−1 phase

To create □□1, the user □□□ uses its own private key □□□□□□ and □□□□□□□□□ as the list of its neighbors' public keys that sent back to it through RRP packets. The user □□□□ creates □□1 on □□□ as follows where □□□ is the concatenation of all received RRPs (note that, each type of ring signature can be used. But we apply the same ring signature that used in the *Reportcoin* project [14]).

First, □□□□ generates a random challenge □ and gets {□□□, □, □, □□ □□□□□, □} where □ is a point on the □(□□), and □□ □□= □□□□□□ □□□□□□□.

Then, □□□□ generates □□□, and □′ and calculates □□1 = (□□□, □□) by □ = 0 : □ − 1 where:

$$\square\square\square_{\square} = \square' \square \square_{\square+1} = h(\square \| \square\square\square_{\square}),$$
$$\square\square\square_{\square+1} = \square\square\square_{\square+1}\square + \square_{\square+1}\square\square\ \square\square_{\square\square_{\square}+1},$$
$$\square_{\square+2} = h(\square \| \square\square\square_{\square+1})$$
$$\square\square\square_{\square+2} = \square\square\square_{\square+2}\square + \square_{\square+2}\square\square\ \square\square_{\square\square_{\square}+2},$$
$$\square_{\square+3} = h(\square \| \square\square\square_{\square+2})$$
$$\vdots$$
$$\square\square\square_{\square-1} = \square\square_{\square-1}\square + \square_{\square-1}\square\square\ \square\square_{\square\square_{\square}-1},$$

$\square_{\square} = h(\square \| \square\square\square_{\square-1})$ and $\square\square\square_{\square} = \square - \square_{\square}\square\square\square_{\square\square_{\square}}$. Therefore, □ = □ □ = □□□□□□ + □□□□□□□□ □. Finally, □□□ sets □□1 = {□, □□, □□ □□□□□}.

5.1.5. Run TSSP phase

The □□□ executes the □□□□ function to get □ 3-tuples as {□□□, □□□□, □□□□} ← □□□□(□, □□□, □□□□□) where □□□ = □□1. Then, each part of the □□□□ function is sent to each □□□□□□□'s public key/address.

5.1.6. Reconstruction and verification−1 on VBC phase

Each □□□□□□□ is obtained by executing (□□□□□□□, ⊥) ← □□□□□(□□□□□, □□□□□□□□) using □ number of □□□'s private keys □□□□□. To do it, VBC members cooperate together to reconstruct the □□1.

Table 6
VBC establishment.

Algorithm 3: VBC - User registration

```
function VBCreg (oldLIDi, oldpuLIDi, LIDi, puLIDi) {
if LEN ≠ 0 then
// It means there is a LIDi in the VBC, therefore update it. Otherwise, add a new parameter for the VBC. (VBC[i].LIDi == oldLIDi;) Update the user
VBC[i].LID = LID;
VBC[i].puLID = puLID;
return 1;
}
else {
// First time
LEN+;
VBC[LEN].LID = LID;
VBC[LEN].puLIDi = puLIDi;
return 1;
}
}
```

Table 7
User registration by VBC members.

Algorithm 4: BC - User registration

```
// This algorithm can be executed by each VBC member
function newUreg (oldUIDi, UIDi, oldpuUIDi, puUIDi) {
if LIDi ≠ msg.sender then
return 0;
// VBC members can update BC members.
else {
// It is similar to Table 5. The main difference between this and Table 5 is to check the "LIDi ≠ msg.sender"
len+;
BC[len].UIDi = UIDi;
BC[len].puUIDi = puUIDi;
return 1;
}
}
```

Table 8
User revocation by VBC members.

Algorithm 5: BC - User revoke

```
// This algorithm is executed by each VBC member.
function Urev (UIDi, puUIDi) {
if LIDi ≠ msg.sender then
return 0;
// It means just LIDis can revoke BC members (after decision-making level)
else {
// Now, VBC member wants to revoke UIDi.
if Exist(BC[i].UIDi == UIDi) then {
Release(BC[i]);
for; i < len i++;
BC[i] = BC[i+1];
len - -;
return 1;
}
}
```

After □□1 reconstruction, each □□□ can verify the reconstructed □□1 = {□, □□□, □, □□ □□□□ } as follow.

First, □□□□ computes □1 = ℎ(□ ||□□□□) and □□□1 = □□1 + □1□□ □□□□ 1. It then compares □□□1 with the corresponded RRP. It repeats this process for all □2, □□□2, ⋯ □□, □□□□ and if all □s are true, □□1 is verified, and each □□□ can submit the verified □□ including □□□s on the BC.

5.1.7. *Ring signature−2 and submit on BC phase*

To submit the valid □□□ on the BC, each □□□□ has to create a valid □□2 using its private key □□□□□ and the list of VBC members' public keys □□□□ □□□. To create □□2, □ □□□ generates random challenge □′ and gets {□□□, □, □ , □□ □□□□ , □′} where □□ □□□□ = □□□□□ □

The continuation of the process is similar to Section 5.1.4 and □□□□ cre- ates □□2.

After the creation of □□2, □□□ can submits □□2 = {□, □□□, □′, □□ □□□□ } on the public BC, where □ is the number of VBC members and □□ □□□□□ is the verification list.

5.1.8. *Verification−2 phase*

After □□2 submission on the BC, each □□□ can find □□2 = {□, □□ □′, □□ □□□□}. All □□□ can verify □□2 on □□□ using the verification algorithm used in Section 5.1.6 with □□□□□□□ as the list of VBC members' public keys. Then, □□□□ obtains □□□ from □□□.

5.2. *Security analysis*

In this section, we analyze the *ARSnCA* protocol and show that it achieves the defined goals presented in Section 3.3.3 in a provable security form.

5.2.1. *User untraceability*

In the *ARSnCA* protocol, A should not be able to trace □) the BC member who executes the □ □□□ function more than one time and □ the VBC member who submits □□2 on the BC more than one time.

Theorem 1. *The ARSnCA protocol provides user untraceability if the* □ □□□ *function and the used ring signature are secure.*

Proof. To find the mentioned links for tracing user(s) who sent more than one report, A designs the two experiments. *First*, an experiment to find a link between the BC member who executed □ □□□ for several times (trace BC member). *Second*, an experiment to find a link between the VBC member who submitted a □□2 on the BC for several times (trace VBC member). The two mentioned experiments are shown in Table 9 (Algorithm 6) and Table 10 (Algorithm 7), respectively.

Table 9
Breaking BC members untraceability.

Algorithm 6 □□□ $\mathcal{A}^{\text{□□-□□□□□□□□□□□□}}$

Input: □, □□□□$_{□□}$, □□□□$_{□□}$, random oracle (signature and □ □□□ oracles)
Output: 1 (if $\mathcal{A}$ traces the BC member) / 0 (if $\mathcal{A}$ cannot trace the BC member)
Goal: Trace □$_{□□,}$
Setup
The Challenger executes *system setup* and *find neighbors* algorithms and gives the set of system public parameters □□□□□□ and RRPs to $\mathcal{A}$.
Experiment
// In this experiment, $\mathcal{A}$ users hybrid experiments.
1. The $\mathcal{A}$ collects and stores enough RRPs, by eavesdropping public channel.
// There is no need to verify received RRP since $\mathcal{A}$ executes this experiment.
2. The $\mathcal{A}$ submits polynomially bounded numbers of □□□$_{□□,\mathcal{A}}$ and □□□□$_{□□,}$ to the signature oracle simulator as queries and collects all returned □□1*s as responses
// The random oracle's outputs/responses are generated like □□1 presented in Section 5.1.4.
// This oracle enables $\mathcal{A}$ to modify the inputs of next oracle.
3. The $\mathcal{A}$ submits the received responses □□1* to the simulator of the □ □□□ as queries, and collects all responses flows of the □ □□□ oracle.
// The two above steps mean that the $\mathcal{A}$ sends polynomially bounded numbers of queries to the random oracle simulators. The random oracle, using the secret values, executes the signature and □ □□□ algorithms to make responses for $\mathcal{A}$. Finally, the random oracles return the real flows of the □ □□□ to $\mathcal{A}$ as the responses. At the end of experiment phase, $\mathcal{A}$ stores the query-response table {□□□$_{□□}$ ‖$_{\mathcal{A}}$□□□□$_{□□}$ ‖ Output[□ □□□]} to use in the guess phase for guessing ideal values.
Challenge
The Challenger generates two random challenges □$_0$ and □$_1$, and creates two ring signatures □□1$_0$ and □□1$_1$. It then executes the □ □□□ function for two times on the two generated ring signatures □□1$_0$ and □□1$_1$. The challenger finally gives □$_0$ and □$_1$, and generated flows (the □ □□□ function's outputs) related to □$_□$ where □ ∈$_□$ {0, 1} to $\mathcal{A}$.
// The $\mathcal{A}$ receives the set {□$_0$, □$_1$, □□□$_□$, □□□$_□$} as the challenge.
Guess
if
The $\mathcal{A}$ guesses the valid □ ∈ {0, 1} for □$_□$ such that □□[(□□□$_□$, □□□$_□$, □$_{□□□□□□□}$) ← □ □□□ (□, □$_□$, □□□□$_{□□}$)| 1 ←*RVerif*(□□1*, □□□□$_{□□}$, □$_□$)] > □ (the outputs of the □ □□□ function are the same outputs were given by the challenger),
Algorithm 6 returns 1 (success): $\mathcal{A}$ can trace □$_{□□,}$
else
Algorithm 6 returns 0 (failure): $\mathcal{A}$ cannot trace □$_{□□}$.
□□□ $\mathcal{A}^{\text{□□-□□□□□□□□□□□□}}$: Based on the stored query-response table {□□□$_{□□}$ $_{\mathcal{A}}$‖□□□□$_{□□,}$ ‖ Output[□ □□□]}, the two random challenges □$_0$ and □$_1$, the user's private key, and the knowledge that □□1$_0$ ≠ □□1$_1$ (the security of the used ring signature was proved in [9]), the flows of □ □□□ are not equal. Therefore,
□□□ $_{\mathcal{A}}^{\text{□□-□□□□□□□□□□□□}}$ = □□[□□□ $_{\mathcal{A}}^{\text{□□-□□□□□□□□□□□□}}$] = □□[$\mathcal{A}$(□□1$_0$, □□1$_1$, □□□$_{□}$) = □□1$_0$ ↔ □□□$_{□}$ |□ ∈ {0, 1}] − $\frac{1}{2}$ < □, and we can say that the *ARSnCA* protocol provides BC members untraceability since the used ring signature provides this feature (the security of this feature is reduced to the security of the used ring signature).
Output: 0

Table 10
Breaking VBC members untraceability.

Algorithm 7 □□□ $\mathcal{A}^{\text{□,□□-□□□□□□□□□□□□}}$

Input: □□□□$_{□□}$, □□□, □□2, random oracle
Output: 1 (if $\mathcal{A}$ traces the VBC member) / 0 (if $\mathcal{A}$ cannot trace the VBC member)
Goal: Trace □$_{□□,}$
Setup
The Challenger executes the *system setup* algorithm and gives the set of system public parameters □□□□□□ to $\mathcal{A}$.
Experiment
1. The $\mathcal{A}$ selects polynomially bounded numbers of random challenges □′.
2. On having the random oracle simulator that returns a valid output value for the signature algorithm, $\mathcal{A}$ sends the selected polynomially bounded numbers of □′, □□□, □□□$_{□,\mathcal{A}}$ and □□□□$_{□□}$ to the signature oracle simulator as queries and collects all returned □□2*s as responses (the □□2*, as the signature oracle output, is generated similar to Section 5.1.7 and satisfies the verification algorithm).
3. The $\mathcal{A}$ stores the query-response table {□′‖ □□□‖ □□□$_{□□}$ ‖$_{\mathcal{A}}$ □□□□$_{□□}$ ‖ □□2*} to use in the guess phase.
Challenge
The Challenger generates two random challenges □□□$_{□□_1}$ and □□□$_{□□_2}$, and creates two □□2$_1$ and □□2$_2$ on the message □. The Challenger then gives □□2$_□$ where □ ∈$_□$ {1, 2} to $\mathcal{A}$.
Guess
if
The $\mathcal{A}$ guesses the valid □ ∈ {1, 2} such that |□□[□□2$_□$ ←*RSign*(□, □□□$_{□□}$, □□□□$_{□□}$)|□ ∈ {1, 2}] − $\frac{1}{2}$| ≥ □,
Algorithm 7 returns 1 (success): $\mathcal{A}$ can trace □$_{□□,}$
else
Returns 0 (failure): $\mathcal{A}$ cannot trace □$_{□□,}$.
□□□ $_{\mathcal{A}}^{\text{□,□□-□□□□□□□□□□□□}}$: Because of using a random challenge □ and the used VBC member's private key □□□$_{□□}$ in □□2, we know □□2* ≠ □□2. Then, as with Algorithm 6 (see Table 9), □□□ $_{\mathcal{A}}^{\text{□,□□-□□□□□□□□□□□□}}$ = □□[□□□ $_{\mathcal{A}}^{\text{□,□□-□□□□□□□□□□□□}}$] = □□[□′* = □′ && □□□$_{□□,}$* = □□□$_{□□,}$] = $\frac{□}{□-1}$ < □. Therefore, the *ARSnCA* protocol is also provides VBC members untraceability since the used ring signature supports this feature.
Output: 0

- *BC members tracing:* To trace the BC member, who sent two reports □□□$_0$ and □□□$_1$, $\mathcal{A}$ has to analyze □□□$_□$ and □□□$_□$ that were sent on the public channel. Algorithm 6, as the mentioned analysis, is shown in Table 9.

  On eavesdropping public channel, $\mathcal{A}$ gets access □ □□□ function' outputs; And $\mathcal{A}$ has to break user untraceability of the used ring signature if it could reconstruct □□1. Based on the proved securities of the □ □□□ function (see Section A.1.3) and used ring signature (see [9]), $\mathcal{A}$ is faced with two main problems. Therefore, it cannot trace the whistleblower.
- *VBC members tracing:* To trace the VBC member, who submitted two different reports □□□ on the blockchain, $\mathcal{A}$ has to analyze the submitted ring signature □□2. Algorithm 7, as the mentioned analysis, is shown in Table 10.

On access to the submitted reports on the blockchain, A has to break the security of the used ring signature, as the main problem, to trace the VBC member who followed/submitted reports. Therefore, the *ARSnCA* protocol provides untraceability for VBC members since the used ring signature is secure.

Based on Table 9 (Algorithm 6) and Table 10 (Algorithm 7), A cannot trace BC members nor VBC members. Therefore, the *ARSnCA* protocol provides user untraceability for both types of users, including BC members and VBC members, since the used □ □□□ function and ring signature are secure.

5.2.2. User anonymity

There are two types of users that A tries to break their anonymity. *First*, a user who wants to keep its identity private is the user who executes □ □□□ (BC members). *Second*, the user who wants to keep its identity confidential is the user who submits the □□□ on the BC (VBC members).

Theorem 2. *The ARSnCA protocol provides user anonymity if the used ring signature is secure.*

Proof. A ring signature is applied as a cryptographic tool/primitive to provide this feature (anonymity) in both types of users. Therefore, to prove the user anonymity in the two mentioned approaches, it should be demonstrated that the used ring signature is worked securely. The anonymity space is the number of members that the user wants to be anonymous among them. In the first type, the anonymity space is □ (based on the *ARSnCA* protocol's details - see Section 5.1.4), and in the second type, the anonymity space is □ (based on the *ARSnCA* protocol's details - see Section 5.1.7). In this section, we generally assume that the anonymity space is □, and a list of members' public keys is assumed □ □□□ ; And we analyze the ring signature with these two assumptions, and we analyze the user pseudonymity in □□ = {□, □□□, □, □}.

Based on the proved security of the used ring signature [9] (or a generic ring signature), the advantage for an outsider A to violate the user anonymity is computed as $\frac{1}{□}$, and the mentioned advantage for an insider A is calculated as $\frac{1}{□-1}$. Therefore, the *ARSnCA* protocol provides user anonymity since the used ring signature is secure.

5.2.3. Resistance against replay attack

The A tries to send one of the previous corresponded messages to receivers for misleading them. This section indicates that A wins to implement the replay attack on the *ARSnCA* protocol with a negligible advantage against the challenger.

Theorem 3. *The ARSnCA protocol provides security against replay attack if the used ring signature and □ □□□ function are secure.*

Proof. There are two types of flows in this attack that A can use for implementing the replay attack:

1. *Using RRP flows:* The RRP flows are the packet sending back from BC members to □□□ . As we assumed that the □□□□(□, □) is a secure signature, and we know that this signature type is fresh. Moreover, □□□□ who waits for RRP knows that what was the content of its sent report and if any malicious □□□□ (where □ ≠ □) repeats an old RRP, □□□□ can detect easily that the received RRP is invalid for (RRP over a wrong □□□ or an old RRP).
2. *Using □ □□□ flows:* The message □□□ in the □ □□□ function is assumed a ring signature (called □□1). Therefore, the □ □□□ input is always fresh. Additionally, the □ □□□ function's output freshness is not dependent on the input. The A has to decrypt all flows first if it wants to repeat each □ □□□ flow. Then, it has to reconstructs □□1 and obtains □□□ and finally forges the □□1 and executes the □ □□□ again. We show the mentioned process in Table 11 (Algorithm 8) in more details. In this experiment, we assume A is one of □□□□ (malicious BC member), and it has the ability to control a large number of VBC members (bigger than the threshold □). It is shown that it can reconstruct □□1 successfully (as an assumption to give help to A, we assume that A can solve secret sharing problem). However, according to Table 11, it can implement the replay attack with a negligible probability.

5.2.4. Resistance against malicious insider attack

According to our defined challenge (that was solved by presenting the *ARSnCA* protocol), the weakness of most DMSs is having a party (central authority) whom all network members should trust on. It can do what it wants, such as user revocation or refusing to sign and submit the received □□□ on the BC. The *ARSnCA* protocol works with no CA, and the group of users (VBC members) decides on network policies and network members. Therefore, there is no CA, and the *ARSnCA* protocol is secure against the malicious server attack (single malicious server or being some malicious insiders).

5.2.5. Resistance against DoS and DDos attacks

Generally, the DoS attacks can be implemented in DMSs in the two below ways.

1. *The □□□s is not submitted on the BC by CA (insider DoS):* In the *VBC establishment* phase, □ is determined. Trusted VBC members can reconstruct and obtain □□□: And at least one of them can submit the reconstructed □□□ on the BC if some malicious users joined the VBC and do not cooperate (some malicious insiders cooperate in implementing DoS attack by refusing in □□1 reconstruction). Therefore, the existing malicious insiders cannot prevent the □□□ submission if they are present in the VBC (reporting system).
2. *General DoS or DDoS attacks:* In this kind of DoS attack, A sends a lot of □□□ to the VBC (or executes □ □□□ for several times). Each □□□□ can check the validity of received report by computing □1 = *h*(□‖□□□□) and □□□1 = □□□1□ + □1□□□ □□1. It is clear that the computation of these calculations is faster than executing the □□□□ function, and each □□□□ can verify the received report (valid or invalid) faster than A who executed the □□□□ function, and A cannot implement DoS attack successfully. Moreover, it should be spent a specific time from the executing time of the □ □□□ function in BC-based networks until arriving parts of the □ □□□ function in □□□□'s public keys/address in addition to paying the submission fee by A. Therefore, A cannot send many packets in a limited time.

As we described in Section 3.3.3, dealing with a DDoS attack is harder than dealing with the DoS attack. However, in the *ARSnCA* protocol, if A wants to execute the □ □□□ function from several locations, the VBC has numerous members, and they are stronger than the case that one party as the CA.

5.2.6. Transparency

Like other blockchain-based reporting protocols [14,17,23] and DMSs [8,9,11,12,19], we also choose the BC as the *ARSnCA* protocol's infrastructure (a transparent database). Therefore, all BC members have access the submitted □□□, and all □□□s are accessible for BC members. This feature, as one of the main blockchain features, was implemented in the phase of *Verification−2* of the *ARSnCA* protocol (Section 5.1.8).

6. Comparison and evaluation

In this section, the *ARSnCA* protocol is compared with other reporting protocols and some other similar protocols. The list of used acronyms and notation for this section is shown in Table 12.

6.1. Feature comparison

This subsection compares both general and security features of the *ARSnCA* protocol with other similar protocols.

Replay attack.

Algorithm 8 □□□ □□□□□□ A

Input: □□□□□ where □ = 1 : □ + 1, □□□□□□□□ , □□□□□□□□ , □□□□□ where □ ≠ □

// It is assumed that A has the ability to control VBC members over □.
Output: 1 (if A can implements replay attack) / 0 (if A cannot implement replay attack)

Goal: Implement replay attack or □□1* = □□1

Setup

The Challenger executes the *system setup* algorithm and gives the set of system public parameters □□□□□□ to A.

Experiment

1. The A eavesdrops public channel, for polynomial time, and collects all generated parts of the □ □□□ function.
2. The A reconstructs all □□1 for all eavesdrops.

// It was assumed that A has an ability to solve the secret sharing problem since it has access □□□□□ where □ = 1 : □ + 1 (it has the ability to control VBC members over □).

3. The A extracts all □□□s from all reconstructed □□1.
4. The A selects two random challenges □* and □′*.
5. The A creates a new □□1* using □□□□□ , □□□□□□□ , □*, and □′*.
6. The A executes □ □□□ (□, □□1*, □□□□□□).
7. The A creates and stores a table, including all inputs-outputs, to use in the guess phase.

Challenge

The Challenger generates a new □□1, and gives it to A.

Guess

if

The A guesses the two valid value for □□1* and Output[□ □□□*] such that □□[□□1* == □□1 □ □□□‖* == □ □□□] >□,
Algorithm 8 returns 1 (success): A can implement replay attack.

else

Returns 0 (failure): A cannot implement replay attack.

□□□ A : In this experiment, A, as a malicious BC member, has to guess two valid random parameters. To calculate □□□A, there are three types of □□□A: □)
□□□ A–1 = □□[□□□ A 1 □□□□□□□□□□] < □ , □□) □ A–2 = □□[□□□ A 2 □□□□□□□□□□] < □ , and □□□) □ A–3 = □□[□□□ A □−1 □=□] < 1/□. Therefore, □□□ A □□□□□□ = □ A–1 □□□ A–2 □□□ A–3 = □□1/□2 < □.
Output: 0

Table 12
The List of Acronyms and Notations.

Acronym and Notation	Description
Admin	Administrator
BCB	Blockchain-based
RS	Ring signature
ZKP	Zero-knowledge proof
□□□□□□□□	Cost of bilinear pairing operation
□□□□□□	Cost of power operation
□□□□□□,□	Cost of elliptic-curve point multiplication operation
□□□□□□	Cost of multiplication operation
□□□□□□	Cost of hash function
□□□□□	Execution time of bilinear pairing operation
□□	Execution time of power/exponentiation operation
□□	Execution time of multiplication operation
□□□,□	Execution time of elliptic-curve point multiplication operation
□□	Execution time of hash operation
√	Provides the feature
×	Does not provide the feature
-	Is not checkable
□	Number of users in ring signature (= 10)

6.1.1. General features

In this section, we briefly compare the general features of *ARSnCA* protocol with some other blockchain-based protocols including ARS-PS [8], KMS [9], e-voting [11], EHR [12], Reportcoin [14], BB2AR [17], BCPPA [19], and QS-RP [23]; the mentioned comparison is shown in Table 13. The main advantage of the *ARSnCA* protocol over other mentioned protocols is that it has no central authority and is secure against the malicious server and privileged insider attacks. In addition to that, in the *ARSnCA* protocol, users can report something anonymously, and they can find their report(s) that is/are submitted on the blockchain with no change if some malicious insiders are present in the system.

6.1.2. Security features

This subsection compares the *ARSnCA* protocol with others in the security aspect; the mentioned comparison is shown in Table 14. As aforementioned, the main security feature that the *ARSnCA* protocol provides is resistance against DoS and DDoS attacks if several privileged insiders are present in the system. Another bold feature of the *ARSnCA* protocol is preserving privileged insiders' privacy so that they can follow the sent report until submission anonymously. The rest of Table 14 discusses the comparison of common attacks on the *ARSnCA* protocol with others.

6.2. Performance

In this section, the *ARSnCA*'s performance will be evaluated and compared with other blockchain-based reporting/similar protocols. As the *ARSnCA* protocol is mainly a reporting protocol, it is compared with those four discussed blockchain-based protocols of ARS-PS (anonymous reporting system) [8], E-voting [11], Reportcoin [14], and BB2AR (anonymous reporting with anonymous rewarding) [17].

A remark should be said that the *ARSnCA* protocol, [14], and [17] applied a ring signature to provide anonymity among several users. A ring signature is a necessary tool for providing anonymity, and the sent report is valid if some witnesses (e.g., 10) have been attached to it. However, in [8] and [11], each customer/voter can generate its rate/vote independent of others.

Table 13
The General Comparison.

Protocol ⇒ Feature ⇓	ARS-PS 2019 [8]	KMS 2020 [9]	E-voting 2018 [11]	EHR 2019 [12]	Reportcoin 2019 [14]	BB2AR 2019 [17]	BCPPA 2020 [19]	QS-RP 2021 [23]	ARSnCA (Our)
Network category	BCB with IDM	BCB with RA	BCB with Admin	BCB with server	BCB with verifier	BCB with authority	BCB with certificate authority	BCB with central authority	BCB with VBC
No central authority	No	No	No	No	No	No	No	No	Yes
Provide reporting	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes
Reporting technique	Short signature	-	-	Signature	RS	RS	-	Encryption	RS
User Anonymity	Conditional	Conditional	Computational	No	Conditional	Conditional	Conditional	Computational	Conditional
User privacy technique	Pseudonym and ZKP	Pseudonym	One-time RS	-	RS	RS	Anonymous certificate	Pseudonym	RS
User untraceability	Conditional	Conditional	Yes	No	Yes	Yes	Yes	Yes	Yes
Transparency	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes

Table 14
The Security Comparison.

Protocol ⟹ Feature ⇓	ARS-PS [8]	KMS [9]	EHR [12]	Reportcoin [14]	BB2AR [17]	BCPPA [19]	QS-RP [23]	ARSnCA (our)
Insiders anonymity and untraceability	×	×	×	×	×	×	×	√
User anonymity and untraceability	√	√	×	√	√	√	√	√
Replay attack security	×	√	√	×	√	√	√	√
DoS-resistance and malicious server resistance	×	×	×	×	×	×	×	√
Message integrity	√	√	√	√	√	√	√	√

6.2.1. Computation and timing

The computational cost and execution time for the four mentioned protocols and the *ARSnCA* protocol are calculated in this section. At first, their computations and execution times are written and calculated, and then the discussion will be presented.

- **Reportcoin** [14]: To submit reports, each user has to do three steps: □) collects witnesses, □□) checks the authentication set for finding the needed index, and □□□) generates the ring signature. The main cost in Reportcoin is the computation for creating the ring signature. Therefore, to create the ring signature over the report (in [14], the report is shown with the notation of □□□), the user need to calculate $2\square\square\square\square_{\square\square,\square} + 2\square\square\square\square_{\square}$.
  Based on the mentioned computations on the user side, the execution time for the user is related to the number of existing users (witnesses) in the ring signature, and it is calculated as $2\square\square_{\square\square,\square} + 2\square\square_{\square}$.
- **BB2AR** [17]: According to the anonymous rewarding concept, the user first has to compute its one-time address that needs to $2\square\square\square\square_{\square\square,\square}$. The user then calculates the ring signature by the cost of $(\square + 3)\square\square\square\square_{\square\square,\square} + 2\square\square\square\square_{\square}$.
  Based on the aforementioned costs, the user's side delay is calculated as $(\square + 5)\square_{\square\square,\square} + 2\square_{\square}$.
- **ARS-PS** [8]: To assign linkable rating tokens to retailers anonymously (in this study, we assume the rating tokens equivalent to a report), customers have to do two steps. *First*, generating rating token, and *second*, generating verifying proof. According to Liu et al. [8] (Sections V-D and V-E), the costumers, for the first-mentioned step, need to calculate $3\square\square\square\square_{\square\square\square\square} + 6\square\square\square\square_{\square} + 5\square\square\square\square_{\square\square,\square} + \square\square\square\square_{\square}$. For the second step, each customer has to generate the proof under the cost of $2\square\square\square\square_{\square\square\square\square} + 13\square\square\square\square_{\square} + 8\square\square\square\square_{\square\square,\square} + 3\square\square\square\square_{\square}$.
  Unlike ring signature-based protocols, each user's calculation and execution time are independent from the number of existing customers/users □. Therefore, each customer has an independent-to-□ delay equal to $5\square_{\square\square\square\square} + 19\square_{\square} + 13\square_{\square\square,\square} + 4\square_{\square}$.
- **E-voting** [11]: As the ElGamal cryptosystem has been applied in the Wang et al.'s e-voting protocol; and based on the ElGamal cryptosystem and the presented e-voting protocol, $3\square\square\square\square_{\square} + 4\square\square\square\square_{\square\square,\square}$ is needed in the user/voter side.
  Similarly, the user's execution time for its vote is determined as $3\square_{\square} + 4\square_{\square\square,\square}$.
- ***ARSnCA*:** As we apply the ring signature used in Reportcoin, the cost and the execution time on the user side are the same cost and execution time calculated for Reportcoin ($2\square\square\square\square\square_{\square\square,\square} + 2\square\square\square\square\square_{\square}$ and $2\square\square_{\square\square,\square} + 2\square\square_{\square}$).

According to Mehibel and Hamadouche [39], Tsai and Su [40], we re-write the execution times based on the time complexity of $\square_{\square}$. Therefore, for each above-discussed protocol,s the total execution time on the user/customer/voter side is re-written as Reportcoin: $2\square\square_{\square\square,\square} + 2\square\square_{\square} = 60\square\square_{\square}$, BB2AR: $(\square + 5)\square_{\square\square,\square} + 2\square_{\square} = 29\square\square_{\square} + 147\square_{\square}$, ARS-PS: $19\square_{\square} + 13\square_{\square\square,\square} + 4\square_{\square} = 7941\square_{\square}$, E-voting: $3\square_{\square} + 4\square_{\square\square,\square} = 1616\square_{\square}$, and *ARSnCA*: $60\square\square_{\square}$.

To have a fair comparison, we assume unified users, for all discussed protocols, who use a smartphone with Hisilicon Kirin 925 2.45-GHz processor, Android 4.4.2, and 3-GB memory [41] so that the execution time for typical multiplicative is 0.731 □□ ($\square_{\square} = 0.731\ \square\square$). The total time in the users' sides is shown in Table 15 where □ is the number of users in ring-signature-based protocols.

Based on Table 15 the comparison of execution time is depicted in Fig. 3. To have a numerical comparison, the *ARSnCA* protocol that applied a ring signature is the fastest protocol among ring signature-based protocols and others if the number of users is assumed □ = 5 (generally, the number of users □ between 5 to 10 is used as a practical number for ring signatures, for example in [14]). However, based on our assumption (□ = 10), the *ARSnCA* protocol is faster than other protocols in such a way it is 62% faster than the discussed e-voting protocol [11], and it is 92% faster than ARS-PS protocol [8].

6.2.2. Communication (On-chain and Off-chain)

In this section, the *ARSnCA* protocol's communication overhead, which includes on-chain and off-chain overheads, is discussed and compared with other protocols. The mentioned discussion and comparison are written in the following and shown in Table 16.

Table 15
Comparison of the Total Execution Time in the User/Customer/Voter Side (□□).

Scheme ⇒ Item ⇓	Reportcoin 2019 [14]	BB2AR 2019 [17]	ARS-PS 2019 [8]	E-voting 2018 [11]	*ARSnCA* (our)
Total execution time	43.86□	21.199□ + 107.457	5804	1181	43.86□

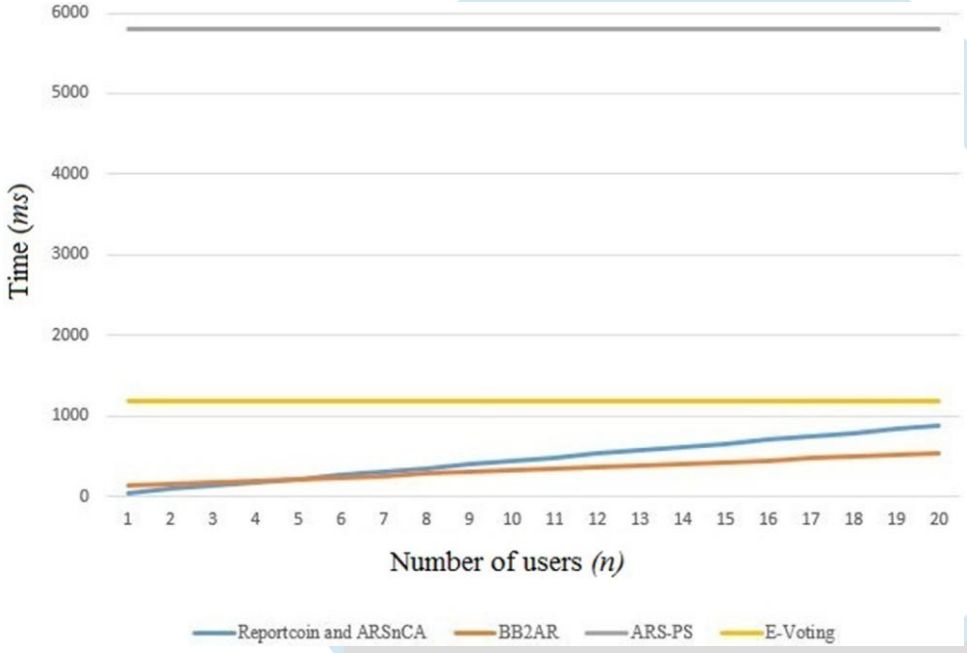

Fig. 3. The Comparison of the total Execution Time (□□).

Table 16
Comparison of Communication (On-chain and Off-chain) Overhead (*parameter*).

Scheme ⇒ Item ⇓	Reportcoin 2019 [14]	BB2AR 2019 [17]	ARS-PS 2019 [8]	E-voting 2018 [11]	*ARSnCA* (our)
Off-chain Overhead	□ + 5	□ + 5	0	0	□ + 5
On-chain Overhead	3□ + 4	0	17	6	3□ + 4

In Reportcoin [14] and the *ARSnCA* protocol, the user first sends the RQP packets to its neighbors (□ + 5) to collect witnesses. Upon receiving witnesses, it creates a ring signature as the report (3□ + 4) and submits the created ring signature on the BC/VBC. The user in BB2AR [17] has no on-chain overhead since it sends a ring signature to the network authority (the network authority submits the verified report on the blockchain). The discussed ARS-PS [8] protocol and E-voting protocol [11] are different since no ring signature has been applied. Therefore, sent parameters are independent of the number of user's neighbors; and 17 and 6 parameters are submitted on the blockchain in ARS-PS (7 parameter as the generated rating token, and 10 parameters as the verified rating token) and E-voting protocol, respectively.

7. Conclusion

This paper introduced a novel challenge in blockchain-based networks and has proposed a solution to address it. The identified challenge revolved around the assumption that the central authority is untrusted. To demonstrate the efficacy of the proposed solution, we present a new blockchain-based protocol named the "Anonymous Reporting System with No Central Authority (*ARSnCA*)." The *ARSnCA* protocol leveraged asymmetric encryption and secret-sharing as cryptographic tools, introducing the concept of a virtual blockchain to circumvent the challenge posed by an untrusted central authority. Notably, the security provided by the employed ring signature in the *ARSnCA* protocol ensures that all users can securely and anonymously submit their reports, guaranteeing their protection against traceability or identification. The introduction of the virtual blockchain concept is expected to enhance network reliability, as critical decisions and submissions are distributed among a group of members rather than relying solely on the central authority within blockchain-based networks.

Future works: The current study employs asymmetric encryption, secret sharing, and ring signatures as key cryptographic primitives to establish a highly reliable reporting protocol. Future research endeavors may explore additional cryptographic primitives to further enhance the protocol's robustness. For instance, employing a weighted secret sharing protocol can be considered to assess and assign weights to VBC members. To bolster report confidentiality, future work might incorporate homomorphic encryption schemes, zero-knowledge proofs, and multi-party computing techniques. Moreover, continual improvements in cryptographic protocols and algorithms could lead to the development of faster and more efficient approaches. Future studies may focus on optimizing existing algorithms or introducing entirely new algorithms to streamline the reporting process. Another avenue for future exploration involves increasing network reliability through the application of threshold secret-sharing protocols, especially in centralized man-

agement systems. This approach could contribute to mitigating risks associated with a single point of control, thereby reinforcing the overall resilience and security of blockchain-based networks.

Ethical Approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Authors' contributions

Saeed Banaeian Far is the study's contributor, and he wrote the first of the manuscript draft. Maryam Rajabzadeh Asaar is the supervisor, and she has reviewed the final version before the submission.

Funding

The authors received no financial support for the research and/or authorship of this article.

Data availability statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Saeed Banaeian Far: Conceptualization, Formal analysis, Methodology, Writing – original draft. **Maryam Rajabzadeh Asaar:** Supervision, Writing – review & editing.

Acknowledgment

We as authors appreciate anonymous reviewers for their insightful comments to improve this work.

Appendix A

A1. The detail of the defined functions

In this section, we describe the details of the two defined functions □ □□□ and □ □ □□□ , and then, for better understanding, we explain other modes of the two functions with numerical examples. Finally, we analyze the security features of □ □□□ and □ □ □□□ in Section A.1.3.

A1.1. The method

We describe the details of our main concept in four phases, including preparation, transaction, verification, and reconstruction in the following (in this section, □ is determined as its minimum value).

1. *Preparation phase:* First, $□_{□□□}$ selects a one-time secret key □□□□□□□_□□□, and divides it into □ parts as □□□□□□□_□□□ = □□□$_1$ ...|| □□□$_□$. Then, $□_{□□□}$ divides □□□□□□□ into □ parts as □□□□□□□ = □□□□□□□$_1$ ... ||□□□$_□$ (we assume the length of the □□□□□□□ is bigger than □) and selects random number $□_{□□□□}$, and current time $□_{□□□□□□□□}$ and calculates the following values.
   □□□$_1$ = h(□□□□□□□$_1$||$□_{□□□□}$ ||$□_{□□□□□□□□}$)
   □□□$_2$ = h(□□□□□□□$_2$||$□_{□□□□}$ ||$□_{□□□□□□□□}$)
   ⋮
   □□□$_{□-1}$ = $□_{□□□□}$ (□□□$_{□-1}$||$□_{□□□□}$ ||$□_{□□□□□□□□}$)
   □□□$_□$ = $□_{□□□□}$ (□□□$_□$ ||$□_{□□□□}$ ||$□_{□□□□□□□□}$)
   where $□_{□□□□□□□_□□□}$(□□□□□□□□) is symmetric encryption algorithm (it is clear that the method to assign each □□□$_□$ to encryption algorithm, is similar to the method of distributing □□□□□□□ will be describe in Section A.1.2 in numerical examples).
2. *Transaction phase:* In this phase, $□_{□□□}$ sends each generated 3-tuple {□□□$_□$, □□□$_□$, $□_{□□□□□□□□}$} to the $□_{□□}$'s public key/address □□□$_{□□}$.
3. *Verification phase:* Upon receiving 3-tuple {□□□$_□$, □□□$_□$, $□_{□□□□□□□□}$} in $□_{□□}$'s public key/address □□□$_{□□}$, each $□_{□□}$ computes □□□$_□$ ||□||$□_{□□□□□□□□□_□□□}$(□□□□□□□□ ||$□_{□□□□}$) = $□_{□□□□□□}$(□□□$_□$) and obtains its part of □□□□□□□_□□□. Then, $□_{□□□}$ s cooperate to reconstruct one-time secret key □□□□□□□_□□□ and executes the symmetric decryption algorithm $□_{□□□□□□□_□□□}$(□□□□□□□□) and obtain □□□□□□□□. Each $□_{□□}$ verifies if □□□ = h(□□□□□□□□||$□_{□□□□}$ ||$□_{□□□□□□□□}$).
4. *Reconstruction phase:* After the verification of all □□□□□□□$_□$ s, all VSC members $□_{□□□}$ cooperate to reconstruct the □□□□□□□.

A1.2. Other modes of the used method

To see other modes of two defined functions and better understanding, we present three numerical examples here. We assume □ = 10, and □ = 100 first, then we check other □.

□ = □□ and □ = □□□: Since we consider □ = 100, all $□_{□□}$ s have to cooperate in the reconstruction process. To create the mentioned equations, $□_{□□□}$ first divides □□□□□□□_□□□ into 10 parts as □□□□□□□_□□□ = □□□□□□□$_1$||□□□□□□□$_2$||... ||□□□□□□□$_8$||□□□□□□□$_9$||□□□□□□□$_{10}$. It then divides the □□□□□□□ into 10 parts as □□□□□□□ = □□□□□□□$_1$||□□□□□□□$_2$||... ||□□□□□□□$_8$||□□□□□□□$_9$||□□□□□□□$_{10}$. The user $□_{□□□}$ selects random number $□_{□□□□}$, and current time $□_{□□□□□□□□}$ and calculates the following values.
   □□□$_1$ = h(□□□□□□□$_1$||$□_{□□□□}$ ||$□_{□□□□□□□□}$)
   □□□$_2$ = h(□□□□□□□$_2$||$□_{□□□□}$ ||$□_{□□□□□□□□}$)
   ⋮
   □□□$_9$ = h(□□□□□□□$_9$||$□_{□□□□}$ ||$□_{□□□□□□□□}$)
   □□□$_{10}$ = h(□□□□□□□$_{10}$||$□_{□□□□}$ ||$□_{□□□□□□□□}$)
   After that, $□_{□□□}$ calculates the following equations.
   □□□$_1$ = $□_{□□□_1}$ (□□□$_1$||$□_{□□□□}$ ||$□_{□□□□□□□_□□□}$ (□□□□□□□$_1$ ||$□_{□□□□}$))
   □□□$_2$ = $□_{□□□_2}$ (□□□$_2$||$□_{□□□□}$ ||$□_{□□□□□□□_□□□}$ (□□□□□□□$_2$ ||$□_{□□□□}$))
   ⋮
   □□□$_9$ = $□_{□□□_9}$ (□□□$_9$||$□_{□□□□}$ ||$□_{□□□□□□□_□□□}$ (□□□□□□□$_9$ ||$□_{□□□□}$))
   □□□$_{10}$ = $□_{□□□_{10}}$ (□□□$_{10}$||$□_{□□□□}$ ||$□_{□□□□□□□_□□□}$ (□□□□□□□$_{10}$ ||$□_{□□□□}$)
   Then, $□_{□□}$ sends each 3-tuples {□□□$_□$, □□□$_□$, $□_{□□□□□□□□}$} to $□_{□□}$ via a public channel.

It is clear, to reconstruct the □□□□□□□, each $□_{□□□}$ has to use its $□_{□□□}$ and obtain its assigned part(s) of □□□□□□□_□□□ to reconstruct the □□□□□□□_□□□. After □□□□□□□_□□□ reconstruction, each $□_{□□}$ can decrypt $□_{□□□□□}$(□□□□□ ||$□_{□□□□}$) and find its assigned part of the original □□□□□□□. Finally, all 10 $□_{□□□}$ cooperate to reconstruct the original □□□□□□□.

All 10 $□_{□□□}$ s can prepare 20 unknown parameters and to reconstruct □□□□□□□ since each $□_{□□□}$ has two unknown parameters (□□□$_{□□□}$ and □□□□□$_□$). In this case, the reconstruction of □□□□□□□_□□□ is similar to the reconstruction of the original □□□□□□□.

□ = □□ and □ = □□: In this mode, we consider □ = 90. Therefore, there is no problem to reconstruct □□□□□□□, if one of the $□_{□□□}$ s be absent. We design the equations such that the 9 parties can reconstruct the original □□□□□□□. To create the equations, $□_{□□□}$ first divides □□□□□□□_□□□ into 10 parts as □□□□□□□_□□□ = □□□$_1$||□□□$_2$||... ||□□□$_8$||□□□$_9$||□□□$_{10}$. Then, it divides the □□□□□□□ into 10 parts as □□□□□□□ = □□□□□□□$_1$||□□□□□□□$_2$||... ||□□□□□□□$_8$||□□□□□□□$_9$||□□□□□□□$_{10}$. The user $□_{□□□}$ selects random number $□_{□□□□}$, and current time $□_{□□□□□□□□}$ and calculates the following values.
   □□□$_1$ = h(□□□□□□□$_1$||□□□□□□□$_2$||$□_{□□□□}$ ||$□_{□□□□□□□□}$)
   □□□$_2$ = h(□□□□□□□$_2$||□□□□□□□$_3$||$□_{□□□□}$ ||$□_{□□□□□□□□}$)

⋮

$□□□_9 = h(□□□□□□□_9 \| □□□□□□_{10} \| □_{□□□□} \| □_{□□□□□□□□})$

$□□□_{10} = h(□□□□□□□_{10} \| □□□□□□_1 \| □_{□□□□} \| □_{□□□□□□□□})$

After that, $□□□_□$ calculates the following equations.

$□□□_1 = □_{□□_1}$ ($□□□_1 \| □□□_2 \| □_{□□□□} \| □_{□□□□□□□\_□□□}$ ($□□□□□□□_1 \| □_{□□□□}$))

$□□□_2 = □_{□□_2}$ ($□□□_2 \| □□□_3 \| □_{□□□□} \| □_{□□□□□□□\_□□□}$ ($□□□□□□□_2 \| □_{□□□□}$))

⋮

$□□□_9 = □_{□□_9}$ ($□□□_9 \| □□□_{10} \| □_{□□□□} \| □_{□□□□□□□\_□□□}$ ($□□□□□□□_9 \| □_{□□□□}$))

$□□□_{10} = □_{□□_{10}}$ ($□□□_{10} \| □□□_1 \| □_{□□□□} \| □_{□□□□□□□\_□□□}$ ($□□□□□□□_{10} \| □_{□□□□}$))

Then, $□□□$ sends each 3-tuples $\{□□□_□, □□□_□, □_{□□□□□□□□}\}$ to $□□□$ via a public channel.

It is clear that to reconstruct the $□□□□□□□$, each $□_{□□□_□}$ has to use its $□□□_□$ and obtain its assigned part(s) of $□□□□□□□\_□□□$ to reconstruct the $□□□□□□□\_□□□$. After $□□□□□□□\_□□□$ reconstruction, each $□□□$ can decrypt $□_{□□□□□}$($□□□□□$ $\| □_{□□□□□}$) and find its assigned parts of the original $□□□□□□□$. Finally, all 9 $□□□_□$ cooperate to reconstruct the original $□□□□□□□$.

All 9 $□□□_□$s can prepare 18 unknown parameters and to reconstruct $□□□□□□□$ since each $□□□_□$ has two unknown parameters ($□□□_{□□_□}$ and $□$). $□ = □□$ and minimum $□$, $□ = □□$ (Minimum threshold): In this mode, we assume the threshold is set as its minimum value (in this case 10%). It means that if one of the $□□□_□$s wants to submit a transaction on the BC, it can do it individually. We design the equations such that each party can reconstruct $□□□□□□□$ individually. To create the equations, $□_{□□}$ uses $□□□□□□□_□□□$and $□□□_{□□□}$ generate needed equations.

First, $□_{□□□}$ divides $□□□□□□□\_□□□$ into 10 parts as $□□□□□□□\_□□□ = □□□_1 \| □□□_2 \| ... \| □□□_8 \| □□□_9 \| □□□_{10}$. Then, it divides the $□□□□□□□$ into 10 parts as $□□□□□□□ = □□□□□□□_1 \| □□□□□□□_2 ... □□□□□□□_8 \| □□□□□□□_9 \| □□□□□□□_{10}$. The user $□□□$ has a random number $□$, and current time $□$ and calculates the following values.

$□□□_1 = h(□□□□□□□ \| □_{□□□□} \| □)$

$□□□_2 = h(□□□□□□□ \| □_{□□□□} \| □)$

⋮

$□□□_9 = h(□□□□□□□ \| □_{□□□□} \| □)$

$□□□_{10} = h(□□□□□□□ \| □_{□□□□} \| □)$

After that, $□$ calculates the following equations.

$□□□_1 = □_{□□□_1}$ ($□□□□□□□\_□□□ \| □_{□□□□} \| □_{□□□□□□□\_□□□}$ ($□□□□□□□ \| □$))

$□□□_2 = □_{□□□_2}$ ($□□□□□□□\_□□□ \| □_{□□□□} \| □_{□□□□□□□\_□□□}$ ($□□□□□□□ \| □_{□□□□}$))

⋮

$□□□_9 = □_{□□□_9}$ ($□□□□□□□\_□□□ \| □_{□□□□} \| □_{□□□□□□□\_□□□}$ ($□□□□□□□ \| □$))

$□□□_{10} = □_{□□□_{10}}$ ($□□□□□□□\_□□□ \| □_{□□□□} \| □_{□□□□□□□\_□□□}$ ($□□□□□□□ \| □$))

Then, $□□□$ sends each 3-tuples $\{□□□_□, □□□_□, □_{□□□□□□□□}\}$ to $□□□$ via a public channel.

It is clear, each VBC member can recover $□□□□□□□$ using $□□□□□□□\_□□□$ as $□□□□□□□ \| □_{□□□} = □_{□□□□□□□\_□□□}$ ($□□□□□□□ \| □_{□□□}$). Therefore, there is no need to cooperate with other $□□□_□$s

A1.3. Security analysis

At first, we show the $□□□□$ function has no information leaks if the presented participants are less than threshold $□$. Then, due to the described definitions of some of network attacks in Section 3.3.3, we analyze the proposed $□□□□$ ($□$, $□□□$, $□□□□_{□□}$) function against some network attacks below.

Information Leak: TSSPs have to be designed such that if A eavesdrops a public channel or controls $100 - □$ % of participants, it learns no information. We analyze this situation in the two cases below.

- *Case 1 (eavesdropping public channel):* The A eavesdrop all 3-tuples $\{□□□_□, □□□_□, □_{□□□□□□□□}\}$ through the public channel. But it learns no information since to obtain $□□□$ ($□□□□□□□ \| □□□□$) it needs to have $□□□_{□□_□}$. Because of the security of one-way hash function $h(.)$, the other eavesdropped parameter $□□□_□ = h(□□□□□□□_□ \| □_{□□□□} \| □_{□□□□□□□□})$ leaks no information.
- *Case 2 (controlling participants):* Each malicious $□□□_□$, as A, receives 3-tuple $\{□□□_□, □□□_□, □_{□□□□□□□□}\}$ in its public key/address, and decrypts $□□□_□$ $\| □_{□□□□} \| □_{□□□□□□□\_□□□}$ ($□□□□□ \| □_□$ $□_{□□□□}$) which leaks no information about $□□□□□□□_□$ befor reconstruction of the symmetric key $□□□□□□□\_□□□$. Therefore, each malicious $□□□$ (or $100 - □$ % of malicious $□□□$s) can release no information about the $□□□□□□$ since they have not enough to decrypt $□_{□□□□□□□\_□□□}$ ($□□□□□□□_□ \| □_{□□□□}$).

Cipher Distinguishability Attack: In this attack, A tries to distinguish between two outputs of $□□□□$ ($□$, $□□□$, $□□□□_{□□}$) functions on two different messages $□□□_1$, and $□□□_2$.

Goal: Distinguishing between the two 3-tuples $\{□□□_{1,□}, □□□_{1,□}, □_{1,□□□□□□□□}\}$ and $\{□□□_{2,□}, □□□_{2,□}, □_{2,□□□□□□□□}\}$ on two messages $□□□_1$, and $□□□_2$

- *Setup:* The Challenger gives the set of system public parameters $□□□□□□$, $□$, $□□□□_{□□_□}$, $□_{□□□□□□□□}$, and the function $□□□□$ to A. This game returns 1 if A can distinguish between two 3-tuples output on two messages $□□□_1$, and $□□□_2$. Else, it returns 0.
- *Experiment:* The A sends polynomially bounded numbers $□□□^*$ to the random oracle as queries. For each query, the random oracle returns $\{□□□^*, □□□^*, □_{□□□□□□□□}\}$ to A such that $\{□□□^*, □□□^*, □$ $\{□□□, □□□^*\} ← □□□□ (□, □□□^*, □□□□_{□□})$. The A stores the query-response $\{□□□, □□□^*\}$ to use in the guess phase.
- *Challenge:* The Challenger creates two 3-tuples $\{□□□_{1,□}, □□□_{1,□}, □_{1,□□□□□□□□}\}$ and $\{□□□_{2,□}, □□□_{2,□}, □_{2,□□□□□□□□}\}$ on two messages $□□□_1$ and $□□□_2$ and gives the two generated 3-tuples and the two messages to A. The challenger asks A to link a 3-tuples to the correct message $□□□_1$ or $□□□_2$.
- *Guess:* To win this game, A has to guess the valid value $□□□^* ∈ \{□□□_1, □□□_2\}$ such that $\{□□□_{1,□}, □□□_{1,□}, □_{1,□□□□□□□□}\} ← □□□□ (□, □□□^*, □□□□_{□□})$.

To guess the valid $□□□^* ∈ \{□□□_1, □□□_2\}$ such that $\{□□□_{1,□}, □□□_{1,□}, □_{1,□□□□□□□□}\} ← □□□□ (□, □□□^*, □□□□_{□□})$ with probability more than $\frac{1}{2}$, A has to select a valid value for $□^*_{□□□□}$ in the created queries. Therefore, $□□□_{A^{h□□_□□□□□□□□□□h□□□□□□□□_□□□□□□}} = □[□□□^* ∈ \{□□□_1, □□□_2\} | \{□□□_{1,□}, □□□_{1,□}, □_{1,□□□□□□□□}\} ← □□□□ (□, □□□^*, □□□□_{□□})] < \frac{1}{2}$ and $|□[□^*_{□□□□} = □_{□□□□}] - \frac{1}{2}| < \frac{1}{2}$. Because of $□^*_{□□□□} ∈ □ [0, 2^{160}]$. Then, $□□□_{A^{h□□_□□□□□□□□□□h□□□□□□□□_□□□□□□}} = □[□^*_{□□□□} = □_{□□□□}] = \frac{1}{2^{160}} < □$. Finally, this game returns 0 and A fails in this game, and $□□□□$ function is secure against cipher distinguishability attack.

Replay Attack: In this attack, A wants to generate the valid $□$ 3-tuples $\{□□□^*, □□□^*, □^*_{□□□□□□□□}\}$ on the same message $□□□$, and sends them again to $□□□_{□□□}$.

Goal: Generating the valid $□$ 3-tuples $\{□□□_□, □□□_□, □_{□□□□□□□□}\}$ on $□$ such that the generated $□$ 3-tuples satisfy the outputs of $\{□□□^*, □□□^*, □_{□□□□□□□□}\} = \{□□□_□, □□□_□, □\}$

- *Setup:* The Challenger gives the set of system public parameters $□□□□□□$, $□$, $□□□□_{□□_□}$, $□_{□□□□□□□□}$, function $□□□□$, and $□□□$ to A. This game returns 1 if A can generate the valid $□$ 3-tuples $\{□□□^*, □□□^*, □_{□□□□}\}$ on $□□□$. Else, it returns 0
- *Experiment:* The A sends polynomially bounded numbers of $□□□$ to the random oracle as queries. For each query, the random oracle returns $\{□□□^*, □□□^*, □_{□□□□□□□□}\}$ to A such that $\{□□□^*, □□□^*, □_{□□□□□□□□}\}$ $\{□□□, □□□^*\} ← □□□□ (□, □□□^*, □□□□_{□□})$. The A stores the query-response $\{□□□, □□□^*\}$ to use in the guess phase.
- *Challenge:* Challenger generates 3-tuples $\{□□□_□, □□□_□, □_{□□□□□□□□}\}$ on $□□□'$, and gives the generated 3-tuples $\{□□□_□, □□□_□, □_{□□□□□□□□}\}$ to A, and asks it to guess valid tuples.
- *Guess:* To win this game, A has to guess the valid value for $□^*$ and generate the valid $□$ 3-tuples $\{□□□^*, □□□^*, □^*_{□□□□□□□□}\}$ as outputs of the $□□□□$ ($□$,$□□□$, $□□□□_{□□}$) on $□□□$ such that the generated $□$

3-tuples {□□□*, □□□*, □□□□□□□□} by A are equal to the generated□ 3-tuples {□□□□, □□□□, □□□□□□□□} that were generated by Challenger.

To hold {□□□*, □□□*, □□□□□□□□} = {□□□□, □□□□, □□□□□□□□}, A has to select the same value for □□□□□ which was selected by Challenger. Because of □* ∈□ [0, 2^160]. Then, □□□ [□□□□□□□□=□□□□□□□□] = □ [□□□□□□□□ = □□□□□] = 1/2^160 < □. Therefore, this game returns 0, and A fails to implement the replay attack successfully.

A2. The registration phase of the Tao et al.'s protocol [9]

In this section, we describe the registration phase of the Tao et al.'s protocol [9] with this paper's syntax since the key distribution of the *ARSnCA* protocol is similar to this. This phase is written below.

The registration phase is executed by CA. Then, □□□□ s take their keys through a secure channel. The user □□□□ takes its key as following.

Each □□□□ sends its identity to CA for registration. The CA checks whether the identity of □□□□ exists. If so, CA aborts the registration. Else, CA selects a random number □□ for □□□ and computes □□ = □□□, □□ = □□□(□□□ ‖□□), and □□□□ = □□ □. The □□□□ 's public key is calculated as □□□□□□ = □□□□□□ □. Finally, CA sends □□, □□, and □□□□□ to □□□ through secure channel and publish □□□□ 's public key □□□□□□□.

References

[1] S. Nakamoto, Bitcoin: a peer-to-peer electronic system, 2008.

[2] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, H. Wang, Blockchain challenges and opportunities: a survey, Int. J. Web Grid Serv. 14 (4) (2018) 352–375.

[3] T. Sermpinis, G. Vlahavas, K. Karasavvas, A. Vakali, DeTRACT: a decentralized, transparent, immutable and open PKI certificate framework, Int. J. Inf. Secur. 20 (4) (2021) 553–570.

[4] I.-C. Lin, T.-C. Liao, A survey of blockchain security issues and challenges, IJ Netw. Secur. 19 (5) (2017) 653–659.

[5] T. Salman, M. Zolanvari, A. Erbad, R. Jain, M. Samaka, Security services using blockchains: a state of the art survey, IEEE Commun. Surv. Tutor. 21 (1) (2019) 858–880, doi:10.1109/COMST.2018.2863956.

[6] G. Wood, Ethereum: a secure decentralised generalised transaction ledger, in: Ethereum project yellow paper, 2014, pp. 1–32. 151.2014

[7] https://ethereum.org/learn/#proof-of-work-and-mining.

[8] D. Liu, A. Alahmadi, J. Ni, X. Lin, X. Shen, Anonymous reputation system for IIot-enabled retail marketing atop pos blockchain, IEEE Trans. Ind. Inf. 15 (6) (2019) 3527–3537, doi:10.1109/TII.2019.2898900.

[9] F. Tao, D. Zhao, Y. Hu, Z. Zhou, Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure, IEEE Trans. Ind. Inf. 16 (3) (2020) 1984–1992, doi:10.1109/TII.2019.2936278.

[10] M. Atzori, Blockchain technology and decentralized governance: Is the state still necessary?, 2015. Available at SSRN 2709713.

[11] B. Wang, J. Sun, Y. He, D. Pang, N. Lu, Large-scale election based on blockchain, Procedia Comput. Sci. 129 (2018) 234–237, doi:10.1016/j.procs.2018.03.063. ISSN 1877-0509

[12] F. Tang, S. Ma, Y. Xiang, C. Lin, An efficient authentication scheme for blockchain-based electronic health records, IEEE Access 7 (2019) 41678–41689, doi:10.1109/ACCESS.2019.2904300.

[13] T.F. Lee, H.Z. Li, Y.P. Hsieh, A blockchain-based medical data preservation scheme for telecare medical information systems, Int. J. Inf. Secur. 20 (4) (2021) 589–601.

[14] S. Zou, J. Xi, S. Wang, Y. Lu, G. Xu, Reportcoin: a novel blockchain-based incentive anonymous reporting system, IEEE Access 7 (2019) 65544–65559, doi:10.1109/ACCESS.2019.2915956.

[15] S. Mettler, The Submerged State: How Invisible Government Policies Undermine American Democracy, University of Chicago Press, 2011.

[16] https://borgenproject.org/types-of-government-systems/.

[17] H. Wang, D. He, Z. Liu, R. Guo, Blockchain-based anonymous reporting scheme with anonymous rewarding, IEEE Trans. Eng. Manage. 67 (4) (2020) 1514–1524, doi:10.1109/TEM.2019.2909529.

[18] H. Wang, Q. Wang, D. He, Q. Li, Z. Liu, BBARS: blockchain-based anonymous rewarding scheme for V2G networks, IEEE Internet Things J. 6 (2) (2019) 3676–3687, doi:10.1109/JIOT.2018.2890213.

[19] C. Lin, D. He, X. Huang, N. Kumar, K.-K.R. Choo, BCPPA: a blockchain-based conditional privacy-preserving authentication protocol for vehicular ad hoc networks, IEEE Trans. Intell. Transp. Syst. (2021), doi:10.1109/TITS.2020.3002096.

[20] G. Fuchsbauer, M. Orrú, Y. Seurin, Aggregate cash systems: a cryptographic investigation of mimblewimble, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Cham, 2019, pp. 657–689.

[21] M. Raya, J.-P. Hubaux, Securing vehicular ad hoc networks, J. Comput. Secur. 15 (1) (2007) 39–68.

[22] X. Lin, R. Lu, C. Zhang, H. Zhu, P.-H. Ho, X. Shen, Security in vehicular ad hoc networks, IEEE Commun. Mag. 46 (4) (2008) 88–95.

[23] B. Far, Saeed, R. Asaar, Maryam, A blockchain-based quantum-secure reporting protocol, Peer-to-Peer Netw. Appl. 14(5) (2021) 2992–3011. 10.1007/s12083-021-01152-z

[24] M. Abe, M. Ohkubo, K. Suzuki, 1-out-of-n signatures from a variety of keys, in: Zheng, Y. (Ed.), Advances in Cryptology - ASIACRYPT 20022002. Lecture Notes in Computer Science, Vol. 2501, Springer, Berlin, Heidelberg, 10.1007/3-540-36178-2-26.

(6) (1976) 644–654, doi:10.1109/TIT.1976.1055638.

[25] W. Diffie, M. Hellman, New directions in cryptography, IEEE Trans. Inf. Theory 22

[26] T. Elgamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. Inf. Theory 31 (4) (1985) 469–472, doi:10.1109/TIT.1985.1057074.

[27] R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Commun. ACM 21 (2) (1978) 120–126.

[28] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613.

[29] T.P. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, in: Annual International Cryptology Conference, Springer, Berlin, Heidelberg, 1991, pp. 129–140.

[30] M. Carpentieri, A perfect threshold secret sharing scheme to identify cheaters, Des. Codes Crypt. 5 (1995) 183–187, doi:10.1007/BF01388382.

[31] P.M. Fathimal, P.A.J. Rani, Threshold secret sharing scheme for compartmented access structures, in: In Cryptography: Breakthroughs in Research and Practice, IGI Global, 2020, pp. 438–448.

[32] A.K. Biswas, M. Dasgupta, Two polynomials based (t, n) threshold secret sharing scheme with cheating detection, Cryptologia 44 (4) (2020) 357–370.

[33] A. Kanso, M. Ghebleh, An efficient (t,n)-threshold secret image sharing scheme, Multimed. Tools Appl. 76 (2017) 16369–16388, doi:10.1007/s11042-016-3917-x.

[34] A. Boldyreva, V. Kumar, Provable-security analysis of authenticated encryption in kerberos, IET Inf. Secur. 5 (4) (2011) 207–219.

[35] M.S. Farash, Security analysis and enhancements of an improved authentication for session initiation protocol with provable security, Peer-to-Peer Netw. Appl. 9 (2016) 82–91, doi:10.1007/s12083-014-0315-x.

[36] J. Ni, K. Zhang, X. Lin, X. Shen, Securing fog computing for internet of things applications: challenges and solutions, IEEE Commun. Surv. Tutor. 20 (1) (2018) 601–628, doi:10.1109/COMST.2017.2762345.

[37] M.A. Khan, K. Salah, IoT security: review, blockchain solutions, and open challenges, Future Gen. Comput. Syst. 82 (2018) 395–411, doi:10.1016/j.future.2017.11.022. ISSN 0167-739X

[38] W. Meng, W. Li, L.T. Yang, et al., Enhancing challenge-based collaborative intrusion detection networks against insider attacks using blockchain, Int. J. Inf. Secur. 19 (2020) 279–290, doi:10.1007/s10207-019-00462-x.

[39] N. Mehibel, M. Hamadouche, Authenticated secret session key using elliptic curve digital signature algorithm, Secur. Privacy 4 (2) (2021) e148.

[40] C.-H. Tsai, P.-C. Su, Multi-document threshold signcryption scheme, Secur. Commun. Netw. 8 (13) (2015) 2244–2256.

[41] V. Kumar, M. Ahmad, A. Kumari, S. Kumari, M.K. Khan, SEBAP: A secure and efficient biometric-assisted authentication protocol using ECC for vehicular cloud computing, Int. J. Commun. Syst. 34 (2) (2021) e4103.