Hindawi
Security and Communication Networks
Volume 2022, Article ID 6841428, 11 pages
https://doi.org/10.1155/2022/6841428

Research Article

Practical Undeniable Multiparty Drawing-Straw Protocol in Asynchronous Networks for Resource-Constrained Information Systems

Ching-Fang Hsu,[1] **Lein Harn**,[2] **Zhe Xia**,[3] **and Hang Xu**[1]

[1]*Computer School, Central China Normal University, Wuhan 430079, China*
[2]*Department of Computer Science Electrical Engineering, University of Missouri, Kansas, MO 64110, USA*
[3]*Department of Computer Science, Wuhan University of Technology, Wuhan 430071, China*

Correspondence should be addressed to Zhe Xia; xiazhe@whut.edu.cn

Received 1 April 2022; Revised 30 April 2022; Accepted 6 May 2022; Published 26 May 2022

Academic Editor: Jinbo Xiong

The next generation of mobile networks and communications (5G networks) has a very strong ability to compute, store, and so on. Group-oriented applications demonstrate their potential ability in resource-constrained information systems (RISs) towards 5G. The security issues in RIS towards 5G have attracted great attention. For example, how to conduct fair and orderly multiparty communication in an intelligent transportation system (ITS). One of the main challenges for secure group-oriented applications in RIS towards 5G is how to manage RIS communications fairly in multiparty applications. In other words, when the users cannot transmit their messages simultaneously, the order of their communication can cause security concerns in multiparty applications. A feasible solution to the problem is for the group of users to follow a specific order to transmit their messages. Otherwise, some users may take advantage over other users if there has no agreeable order to be followed. In this paper, we propose a novel cryptographic primitive, called multiparty drawing-straw (MDS) protocol, which can be used by a group of users to determine the order of the group to participate in the multiparty applications. Our scheme is based on Pedersen's verifiable secret sharing (VSS), which is a well-known scheme. Our proposed protocol is fair since the output is uniformly distributed, and this is an attractive feature for secure multiparty applications in RIS towards 5G.

1. Introduction

Since the emergence of the next generation of mobile networks and communications (5G), technologies such as resource-constrained information systems (RIS) towards 5G have attracted more attention, as various smart devices have been constantly connected to the Internet over the past decades. The number of devices connected to the Internet is increasing since its appearance. Now, this number far exceeds that of people in the world, we are no longer talking about the Internet but about the internet of things (IoT). IoT gives rise to revolutionary applications for emerging technologies of RIS towards 5G. group-oriented applications also show the great potential of society.

Group-oriented applications demonstrate the importance of RIS towards 5G, such as joint data collection for traffic analysis, weather prediction, multiuser interactive computation, and so on. These applications motivate the demand for secure group-oriented applications over open and insecure networks. In particular, the devices in RIS towards 5G are heterogeneous, and the RIS communication environments are asynchronous, where multiple users cannot transmit their messages simultaneously. One of the main challenges for secure group-oriented applications in RIS towards 5G is how to secure the communications among these heterogeneous devices in such an asynchronous environment. Note that it is widely known that asynchronous transmission can cause security problems in cryptographic functions.

Devices in RIS towards 5G generate, process, and exchange vast amounts of security and safety-critical data as well as privacy-sensitive information; hence, they are appealing targets of various attacks [1–8]. To ensure the correct and safe operation of RIS towards 5G systems, it is crucial to ensure the integrity of the underlying devices, in particular of their code and data, against malicious modifications [9]. Recent researches have revealed many security vulnerabilities in the embedded devices [2, 4, 6, 7, 10, 11]. This highlights new challenges in the design and implementation of secure embedded systems that typically must provide multiple functions, security features, and real-time guarantees at a minimal cost [12]. How to design a lightweight protocol to determine the order of the group members to communicate in RIS towards 5G applications is needed in a network that involves multiple devices/users. For example, how to manage multiparty communication in an intelligent transportation system (ITS; see Figure 1). In some specific applications, like multiparty bidding or multiparty gaming, for the sake of fairness among users, users need to transmit their messages in a particular order. Otherwise, users can gain unfair advantages over other users if there has no particular order to be followed. Although users can rely on a mutually trusted center to decide this order, most Internet users would prefer to make their own decisions. The objective of this paper is to design such a lightweight protocol to determine the order of the group members to participate in applications.

In many multiparty applications, the users need to follow a specific order to transmit their messages. Otherwise, messages can collide with each other if multiple transmissions occurred simultaneously. Moreover, in some applications, a user who on purposely transmits his message last may gain unfair advantages. Coin flipping is a simple way of deciding the order between two users. It is widely used in sports and other games to decide the random factors such as which side of the field a team will play from or which side will attack or defend initially. Coin-flipping protocol is a cryptographic primitive that has been introduced by Blum [13] and is one of the basic building blocks of secure two-party computation. Coin flipping is the process of throwing a coin into the air to choose between two possible and equally likely outputs. In cryptography, a commitment scheme can be used to achieve a coin-flipping protocol. Aharonov et al. [14] proposed a quantum protocol with no dishonest player that can bias the coin with a probability higher than 0.9143. Ambainis [15] proposed an improved protocol with cheating probability at most 3/4. Since then, several different protocols have been proposed [16, 17] that achieve the same bound of 3/4. In the following, we describe Blum's two-party coin-flipping protocol [13] between Alice and Bob:

(i) Alice chooses a random bit $a \in \{0, 1\}$ and sends a commitment $c = commit\ (a)$ to Bob

(ii) Bob chooses a random bit $b \in \{0, 1\}$ and sends it to Alice

(iii) Alice sends the bit a to Bob together with *decommit* (c)

(iv) If Bob does not abort during the protocol, Alice outputs $a \oplus b$; otherwise, she outputs a random bit

(v) If Alice does not abort during the protocol and c is a commitment to a, then Bob outputs $a \oplus b$; otherwise, he outputs a random bit

The trend of network applications inspires us to consider scenarios involving multiple players. A novel cryptographic primitive, called *multiparty drawing-straw protocol* (MDS), is introduced in this paper. This technique can be used by a group of users to determine the order of the group to participate in applications. The users need to follow the decided order to take turns to make a movement or release a message in these applications. For example, in multiparty computation, multiple parties want to jointly compute a function over their inputs and keep these inputs private. MDS can be used to determine the order of releasing their inputs. In a real-world solution to provide MDS, the group leader prepares a set of straws of different lengths. Each user of the group randomly draws a straw from the unseen set of straws prepared by the group leader. At the end of the offering, the order of the group is determined by the lengths of straws chosen by group users. The fairness of this solution depends on the trustworthiness of the group leader. If the group leader colludes with any group member, the output of this process can be biased. The requirement of a mutually trusted party is unrealistic in some applications. The MDS without the assistance of a mutually trusted party is desirable.

If there are only two players in MDS, the coin-flipping protocol [13, 18] can be used to determine the order of players. The "*winner*" of a coin-flipping protocol can be the starter. Thus, the coin-flipping protocol is a special type of MDS. Actually, we can use the coin-flipping protocol in a straightforward manner to provide a solution for a general MDS. In this solution, all players are arranged on the leaves of a binary tree. Using a coin-flipping protocol between two users can determine a "*winner*." Repeatedly executing the coin-flipping protocol multiple times following the tree structure (i.e., with complexity $O(n)$ can determine the "1^{st} *winner*" among n users). Then, using the same approach on remaining $n - 1$ users can determine the "2^{nd} *winner*" and so on. However, this approach is not effective because of the time-consuming process. Multiparty coin-flipping protocols [19–21] have been developed recently. However, these protocols are restricted for multiple parties to jointly choose one of the two possible outputs, which are different from our MDS. In 2008, Lit et al. [22] have proposed a secure multiparty ranking problem (SMR) [22], which is extended from Yao's Millionaires' problem [23]. This problem has been studied in [24]. We assume that there are n users and each user has a secret input. In SMR, it intends to get the order of inputs in the ascending ranking sequence while not leaking the value of any input. In particular, each user knows his order of input but does not know the orders of the other users' inputs. The SMR is different from MDS since (a) in SMR each user knows only the order of his input, but in MDS each user knows all inputs, and (b) in SMR the inputs

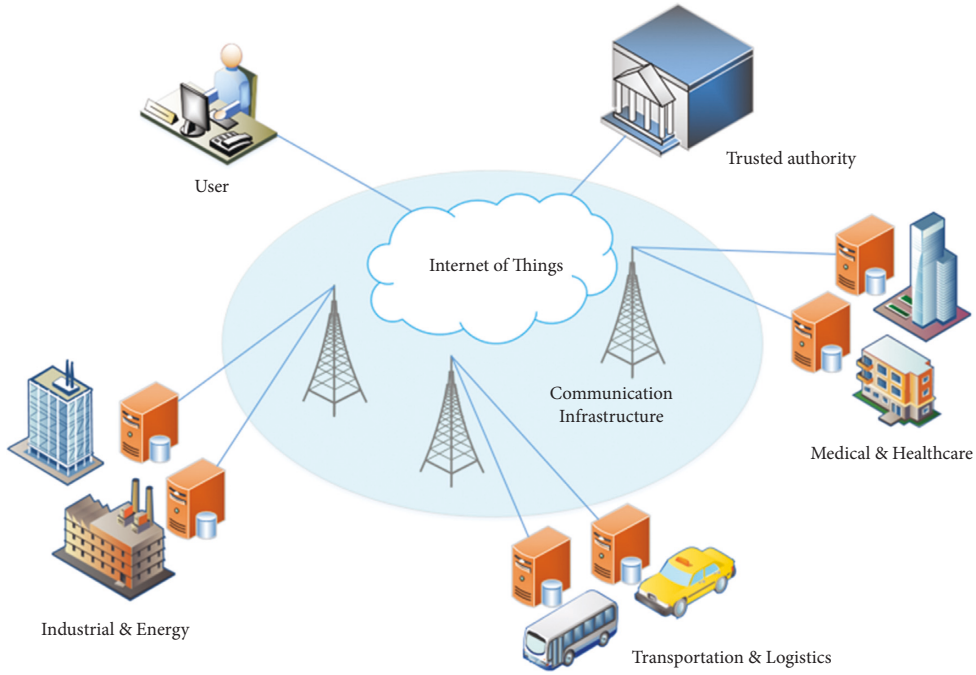

Figure 1: A typical ITS model.

are users' data, but in MDS the inputs are random secrets selected by users.

In this paper, we propose a *multiparty undeniable drawing-straw protocol* (MUDS). Formally speaking, this primitive realizes the following function: $\underbrace{(s_1, s_2, \ldots, s_n)}_{n} \mapsto (o_1, o_2, \ldots, o_n)^n$, where $\underbrace{(s_1, s_2, \ldots, s_n)}_{n}$ are an array including n secret inputs and $(o_1, o_2, \ldots, o_n)$ is a permutation uniformly distributed over $\{1, 2, 3, \ldots, n\}$. User U_i receives his order o_i along with other orders. We present a protocol for this primitive. Our protocol consists of two phases, the commitment phase and the open-commitment phase. In the commitment phase, every group user chooses a secret and releases a commitment of the secret. In the second phase, every group user opens the commitment by revealing the secret. The output of the group order is determined by all released secrets of group users. In our protocol, after making any commitment in phase 1, the user can no longer deny his commitment in phase 2 since we employ a *secret sharing scheme* (SS) and *verifiable secret sharing scheme* (VSS) in our design. If the majority of users are honest, we can prove that our protocol is secure by setting the threshold, t, as $t = [n/2]$, where n is the number of group users.

Many communication networks in practice are asynchronous in which multiparty users cannot transmit their messages simultaneously. The asynchronous transmission can cause security problems in cryptographic functions. For example, the work in [19] has used Blum's two-party coin-flipping protocol [13] to demonstrate the problem. The main concern in designing coin-flipping protocols is to prevent the bias of the output. The bias of a coin-flipping protocol measures the maximum influence of malicious parties on the output of the honest parties. The bias is 0 if the output is always uniformly distributed, and the bias is 1/2 if the adversary can force the output to be always (say) 1. In Blum's protocol [13], since Alice recovers the output $a \oplus b$ before sending her decommit (c) to Bob, [19] demonstrated that Alice has an advantage over Bob, and the bias of the protocol is 1/4.

We use another example, *rational secret sharing* [25], to demonstrate the security problem caused by asynchronous networks. In 2004, Halpern et al. [25] considered a scenario in which users in the secret reconstruction are neither completely honest nor arbitrarily malicious; instead, the users are assumed to be rational. In rational secret sharing, it assumes that all users are rational and want to maximize their utility. A rational user acts honestly when he cannot gain any advantage over other users (i.e., they will all obtain the secret) but acts dishonestly when he can gain an advantage over others (i.e., he is the only one to obtain the secret). The classical SSs including Shamir's SS [26] fail in a rational setting. In Shamir's secret reconstruction, if the user who broadcasts his share last, the user will see that others have broadcasted their shares already. Thus, he will remain silent because other parties cannot reconstruct the secret, but this user can reconstruct the secret by using his own share and shares broadcasted by other parties. There are vast research papers on rational secret sharing (RSS) [27–29]. In fact, the objective of the RSS scheme is to ensure rational users that the secret can be reconstructed successfully. This is the same as that of the fair secret reconstruction scheme, which was originally proposed by Tompa et al. in [30] in 1988. In most RSS schemes, information exchanged among users is restricted to be in a synchronous network. There are only handful papers on RSS schemes assuming asynchronous networks. These include Fuchsbauer et al.'s result [27],

which requires cryptographic primitives, and the results of Ong et al. [28] and Moses et al. [31], which require to assume that a certain number of shareholders must be honest.

The motivation of our paper is to develop a multiparty coin-flipping protocol that involves more than two parties without the assistance of a mutually trusted party. RIS towards 5G devices can employ this protocol to determine their order in Internet applications. Since the order is determined by all RIS towards 5G devices, this protocol needs to prevent dishonest devices from cheating in the process. We consider adopting both Shamir's SS and Pedersen's noninteractive VSS to achieve this objective. The primary reason to adopt both schemes is due to their simplicity to be implemented in an asynchronous network. Shamir's SS is unconditionally secure and polynomial-based, which is the most widely used secret sharing scheme since it is simple and computationally efficient. While for Pedersen's VSS, the privacy of Pedersen's VSS is unconditionally secure, and the correctness of the shares is based on a computational assumption. In Pedersen's VSS, verification is based on the commitments computed by the owner of the secret, and there is no interaction among verifiers during verification. The verification of shares can be performed by each verifier individually.

We propose a novel design to overcome the security problem caused by asynchronous networks. Our protocol is *undeniable* since in the process of making a commitment for the secret, where the secret needs to be shared by all other users. Moreover, shares can be verified by other users. Thus, after making a commitment, the user can no longer deny his commitment. If some user denies making the commitment, the secret can still be able to reconstruct by other honest users. The undeniable feature prevents the users from disrupting the protocol by releasing either a fake secret or no information after making their commitments. Our protocol is based on Shamir's SS [26] and Pedersen's verifiable secret sharing scheme (VSS) [32]. Thus, our design is particularly suitable for group-oriented applications in RIS towards 5G.

Here, we summarize the contributions of our paper.

(i) A novel cryptographic primitive, called *multiparty drawing-straw protocol* (MDS), is introduced

(ii) An MDS protocol with undeniability (MUDS) is proposed that can overcome the security problem caused by asynchronous networks

(iii) MUDS is useful in many multiparty applications, providing a fair way to determine an order of users

The rest of the paper is organized as follows. In Section 2, we present some preliminaries. The model of our proposed protocol is introduced in Section 3, including protocol description, type of entities, and attacks of the proposed protocol. The protocol is outlined in Section 4. We give a concrete protocol in Section 5. The conclusion is given in Section 6.

2. Preliminaries

Our proposed MUDS is based on Pedersen's VSS [33]. We review this scheme in this section.

Chor et al. [34] proposed the notion of VSS in which shareholders can verify that their shares are valid without revealing the secrecy of their shares and the secret. We give a definition of VSS below.

Definition 1 (*t-out-of-n* verifiable secret sharing scheme (VSS)). A *t-out-of-n* verifiable secret sharing scheme $\pi = (G, R, V)$ consists of a sharing algorithm G, a reconstruction algorithm R, and a verification algorithm V. The sharing algorithm G guarantees that it is impossible for any adversary to reconstruct the secret from fewer than t shares. The reconstruction algorithm R guarantees that the secret can be recovered from any t or more than t shares. The verification algorithm V guarantees that shareholders can verify their shares are generated consistently without compromising the secrecy of both their shares and the secret.

VSSs of Feldman [32] and Pedersen [33] are based on cryptographic commitment schemes. The security of Feldman's VSS is on the hardness of solving discrete logarithm, while the privacy of Pedersen's VSS is unconditionally secure, and the correctness of the shares is based on a computational assumption. Benaloh [35] proposed an interactive VSS, which is unconditionally secure. Stadler [36] proposed the first publicly verifiable secret sharing (PVSS) scheme that allows each shareholder to verify the validity of all shares. Most noninteractive VSSs [30, 32] can only verify the validity of his/her own share, but not of other shareholders' shares. The security of Schoenmaker's PVSS [37] is based on the discrete logarithm problem. Peng and Wang's PVSS [38] uses a linear code, and Ruiz and Villar's PVSS [39] uses Pailler's cryptosystem [40]. There are noninteractive PVSSs based on bilinear pairing [41, 42].

Pedersen's VSS is information-theoretic secure. There are public parameters, $g, h \in Z_p$, and assumes that no one knows $\log_g h$. Pedersen's VSS uses the following commitment scheme.

Pedersen's Commitment Scheme. To commit the secret S, the dealer computes and publishes a commitment $E(s,k) = g^s h^k = E_0 \bmod p$, where k is a random integer with $k \in Z_p$. Such a commitment can later be opened by releasing s and k.

Definition 2 (perfectly hiding commitment (PHC) scheme). A commitment scheme is perfectly hiding if it does not reveal any information about the committed value in the commitment phase.

In [33], it has proven that the commitment E_0 reveals no information on the secret S and that the committer cannot open a commitment to s as $s' \neq s$ unless he can solve $\log_g h$. Pedersen's commitment scheme is a PHC. Pedersen's VSS consists of three algorithms as shown in Figure 2.

3. Model

3.1. Description of MUDS. MDS deals with the following setting: n users, $U = \{U_1, U_2, \ldots, U_n\}$, interactively work together to uniformly choose an order for some computation. In the following, we give a formal definition of MDS.

Definition 3 (multiparty drawing-straw (MDS) protocol). MDS computes the following functionality:

Share generation

To commit the secret S, the dealer computes and publishes a commitment $E(s,k) = g^s h^k = E_0 \bmod p$, where k is a random integer with $k \in Z_p$. Then, the dealer chooses two random polynomials, $f(x) = s + a_1 x + \cdots + a_{t-1} x^{t-1} \bmod p$ and $g(x) = k + b_1 x + \quad b_{t-1} x^{t-1} \bmod p$,with degree $t-1$ and $f(0) = s$ and $g(0) = k$. Dealer generates shares, $(f(x_j), g(x_j))$, $j = 1, 2, \quad , n$, of users. The dealer computes and publishes commitments to the coefficients of the polynomials, $f(x)$ and $g(x)$,as $E(a_i, b_i) = g^{a_i} h^{b_i} \bmod p = E_i$, $i = 1, 2, \cdots, t-1$

Share verification

For each pair of shares, $(f(x_i), g(x_i))$, user, U_i, can verify the pair of shares by checking whether $E(f(x_i), g(x_i)) \stackrel{?}{=} \prod_{j=0}^{t-1} E_j^{x_i^j} \bmod p$. If it passes the test, the user is convinced that the pair of share is generated consistently.

Secret reconstruction

For example, when there are j (i.e., $t \le j \le n$) shareholders with their shares, $\{f(x_{i_1}), f(x_{i_2}), \cdots, f(x_{i_j})\}$, the secret can be recovered as $s = f(0) = \sum_{r=1}^{j} f(x_{i_r}) \prod_{v=1, v \ne r}^{j} \frac{-x_{i_v}}{x_{i_r} - x_{i_v}} \bmod p$.

Note, in Figure 2, we use following symbols to denote operations listed in this Figure.

$PHC(s) = E(s,k) \Leftrightarrow \sigma \leftarrow PHC(s)$.

$V(f(x_i, g_i)) \stackrel{?}{=} \prod_{j=0}^{t-1} E_j^{x_i^j} \bmod p \Leftrightarrow V(f(x_i, g_i)) \stackrel{?}{=} 1$.

$s = f(0) = \sum_{r=1}^{j} f(x_{i_r}) \prod_{v=1, v \ne r}^{j} \frac{-x_{i_v}}{x_{i_r} - x_{i_v}} \bmod p \Leftrightarrow s \leftarrow R(f(x_{i_1}), f(x_{i_2}), \cdots, f(x_{i_j}))$.

Figure 2: Pedersen's VSS.

$$\underbrace{(s_1, s_2, \cdots, s_n)}_{n} \mapsto (o_1, o_2, \cdots, o_n)^n, \tag{1}$$

where l is a security parameter, $\underbrace{(s_1, s_2, \cdots, s_n)}_{n}$ are an array including n secret inputs, and $(o_1, o_2, \ldots, o_n)$ is a permutation uniformly distributed over $\{1, 2, 3, \ldots, n\}$. At the end, each user U_i obtains his order o_i along with an order of others.

The output permutation $(o_1, o_2, \ldots, o_n)$ obtained in MDS should be determined by all users without the assistance of a mutually trusted third party. We adopt Pedersen's VSS in our design. In the first phase, each user needs to select a random secret and then compute and release a commitment of the secret to all other users. In the second phase, each user releases his secret to all other users. To avoid the problem caused by any user who may deny releasing his real secret in the second phase, we introduce the following definition of MUDS.

Definition 4 (multiparty undeniable drawing-straw (MUDS) protocol). In a MUDS, no user can deny his secret after revealing his commitment of the secret to others. The security of this functionality is called undeniability.

3.2. Entities and Possible Attacks. The security objective of our protocol is to enable all IoT devices to work together to determine the order among them in a fairway. Since the protocol allows each device to contribute an input and the final order is determined by all inputs, our protocol needs to prevent dishonest devices from cheating in the process. We consider dishonest devices (also called *attackers*) may take advantage of most asynchronous networks by releasing their inputs last. Thus, we divide our protocol into two phases. In the first phase, each input is divided into shares by the device owner. Shares are distributed to other devices secretly. A commitment of this input is also published by the owner.

Other devices can verify that their shares are generated consistently by the owner. In the second phase, each input can be released in any asynchronous way. If any dishonest device owner refuses to release their input or releases a fake input, other honest devices can work together to recover the input. We adopt Shamir's SS and Pedersen's VSS to achieve this objective.

In a VSS, the owner of the secret is the *prover*, and all other users are the *verifiers*. The verifiers want to verify that their shares are generated consistently without compromising the secrecy of the secret. In Pedersen's VSS, verification is based on the commitments computed by the owner of the secret, and there is no interaction among verifiers during verification. The verification of shares can be performed by each verifier individually. Inconsistent shares may be generated due to the following two reasons: (a) in shares generation/distribution, nature noise, such as transmission noise or computational error, may cause the inconsistency and (b) inconsistent shares may be generated by a user who tries to cheat other honest users. In summary, we adopt Pedersen's VSS in our design since Pedersen's VSS is (a) unconditionally secure and (b) noninteractive.

Attackers may try to obtain secrets from commitments. Pedersen's commitment can prevent this attack. Moreover, we need to prevent colluded attacks on users. Since communication networks are asynchronous, colluded attackers can always release their fake secrets after knowing the secrets of other users. Any fake secret can be detected by VSS. Moreover, in our protocol, we employ a threshold SS to ensure that any committed secret can always be reconstructed by the majority of honest users if the threshold is $t = [n/2]$, where n is the number of users.

3.3. Properties. Our protocol has the following properties:

Randomness. The output is uniformly distributed. No user can influence the output.

Secrecy. From the commitment of each secret, the secret cannot be recovered. Furthermore, the secret is protected by a threshold SS.

Efficiency. In our proposed protocol, all users work together to determine the output. We use Shamir's SS [26] and Pedersen's VSS [33] based on polynomials. At the beginning of each phase, each user needs to act as a dealer to compute and release values to others. There has no interaction among users to verify shares. Since both Shamir's SS and Pedersen VSS [33] are simple and efficient, our protocol is very efficient.

Undeniability. After publishing any commitment of the secret, the user can no longer deny the secret since the secret can also be recovered by honest users.

4. Proposed Protocol

4.1. Outline. Let us assume that there are n users, $U = \{U_1, U_2, \ldots, U_n\}$, participated in a MUDS. These users need to interactively work together to generate an output, which is the order of the users. In our proposed protocol, there are two phases, the commitment and open-commitment phases. In the commitment phase, each user selects a secret and acts as the dealer to use a threshold SS to generate shares for other users. Each user makes the commitment of the secret publicly known. For each received share, other users can verify that the share is generated consistently by the owner of the secret. If the verification is failed, a request for regeneration of a share can be sent to the owner of the secret till a share is verifiable.

In the open-commitment phase, each user releases his secret of the commitment to other users. Each released secret can be verified by other users using his commitment made in the commitment phase. If any released secret is an invalid secret, other honest users can work together to recover the secret by using their shares of the secret obtained in the commitment phase. This property, called *undeniability*, in our proposed protocol prevents any user to deny his commitment of a secret.

After obtaining all secrets of users, each user can determine the order of group based on the secrets selected by users.

4.2. Protocol. We illustrate the detail of the protocol in Figure 3.

4.3. Security Proof. We say that two probability ensembles are *statistically indistinguishable* if their statistical difference is negligible.

Lemma 1. *In our framework, if all users are honest, the probability ensemble defined by the order* $(o_1, o_2, \ldots, o_n)$ *and the probability ensemble defined by a permutation uniformly chosen over* $\{1, 2, 3, \ldots, n\}$ *are statistically indistinguishable.*

Proof. We observe that the difference between the two probability ensembles is that some orders probably have the same value in the former. That is, the event $\exists i \neq j (o_i = o_j)$ may appear in the former ensemble. Fortunately, we can show that this event occurs with a negligible probability.

From step 3, we know that $\Pr[\exists i \neq j (o_i = o_j)] = \Pr[\exists i \neq j (\rho_i = \rho_j)]$. If all users are honest, then $\rho_1, \rho_2, \ldots, \rho_n$ are uniformly distributed. We have $\Pr[\exists i \neq j (\rho_i = \rho_j)] \leq \sum_{i \neq j} \Pr[(\rho_i = \rho_j)] \leq (n/2) \times (1/2^l)$.

Thus, we have $\Pr[\exists i \neq j (o_i = o_j)] \leq (n/2) \times (1/2^l)$. Note that $(n/2)$ is a positive constant. Thus, the event $\exists i \neq j (o_i = o_j)$ occurs with a negligible probability.

We say that two probability ensembles X, Y are *computationally indistinguishable*, denoted as $X \approx Y$, if there is no probabilistic polynomial-time algorithm distinguishing them. □

Lemma 2. *In our framework, let* $\overline{\rho}$ *denote the probability ensemble defined by* $\rho_1, \rho_2, \ldots, \rho_n$, *and* $\overline{r}$ *denote the probability ensemble defined by a bit-string uniformly chosen over* $\{0, 1\}^{nl}$. *If the commitment scheme is a PHC, the secret sharing*

Phase 1: Commitment phase

Step 1. Each user U_i uniformly chooses a secret $s_i \in_U \{0,1\}^{nl}$, and computes a commitment $\sigma_i \leftarrow PHC(s_i)$ and publishes σ_i.

Step 2. Each user U_i following a *t-out-of-n* VSS to share his secret s_i i.e., $(\delta_1^i, \delta_2^i, \ldots, \delta_n^i) \leftarrow G_{r_i}(s_i)$, and sends each sub-share δ_j^i to U_j for $j \neq i$. Furthermore, U_i commits to the random integers r_i as $\gamma_i \leftarrow PHC(r_i)$ and broadcasts the commitments γ_i.

Step 3. Each user U_i verifies the legality of the sub-share δ_i^j received from other users U_j, i.e., $V(\delta_i^j, \gamma_j) \stackrel{?}{=} 1$, for $j = 1, 2, \ldots, n$, $j \neq i$. If the verification fails, U_i requests user U_j to re-generate a new sub-share.

Phase 2: Open-commitment phase

Step1. Each user, U_i opens its commitment σ_i to his secret, i.e., broadcasts s_i.

Step 2. Each user, U_i checks de-commitments of other users. That is, verifies $\sigma_j \stackrel{?}{=} PHC(s_j)$, $j \neq i$.

Case1: the verification succeeds. Obviously, U_i obtains the secret s_j of user U_j.

Case2: the verification fails. U_i will work with other honest users to recover the real secret s_j of user U_j. More specifically, each user U_i first releases his own sub-share δ_i^j received from user U_j to all other users. The legality of each released sub-share δ_i^j can be checked by evaluating $V(\delta_i^j, \gamma_j) \stackrel{?}{=} 1$. After obtaining t legal sub-shares, $\delta_{i_1}^j, \delta_{i_2}^j, \ldots, \delta_{i_t}^j$, each U_i can recover $s_j \leftarrow R(\delta_{i_1}^j, \delta_{i_2}^j, \ldots, \delta_{i_t}^j)$.

Step 3. Each user, U_i, computes a bit string $\rho \leftarrow \bigoplus_{j=1}^{n} s_j$ and evenly divides ρ into n strings of bit sequences, $\rho_1, \rho_2, \ldots, \rho_n$. U_i sorts these n strings of bit sequences in a descending order. Let ρ_i be the o_i–th smallest string of bit sequences in the set of strings. Then U_i obtains his order o_i and the orders of other users. Finally, U_i outputs the order $(o_1, o_2, \ldots, o_n)$.

Figure 3: Proposed protocol.

scheme is a t-out-of-n VSS, and there are at most $t-1$ *malicious users, then* $\overline{r} \approx \overline{\rho}$.

Proof. Let us focus on $\overline{\rho}$ and assume that there is at most one malicious user U_i. There are the following types of attacks that can possibly bias $\overline{\rho}$.

(1) The secret S_i chosen by malicious U_i is not uniformly distributed.

Note that $\rho \leftarrow \oplus_{j=1}^{n} s_j$. Thus, U_i cannot bias $\overline{\rho}$, and then $\overline{r} \approx \overline{\rho}$ holds.

(2) Malicious user U_i may refuse to release their decommitment or releases an illegal decommitment.

The *t-out-of-n* VSS guarantees that other honest users can recover the secret. This attack cannot bias $\overline{\rho}$, and then $\overline{r} \approx \overline{\rho}$ holds.

(3) In step 1, phase 2, U_i may open his commitment σ_i to be a maliciously chosen value $\widetilde{S}_i$, which is different from the real value he has committed in step 1 of phase 1.

If U_i succeeds in this cheating, $\overline{\rho}$ is not uniformly distributed. Fortunately, the computationally binding the commitment guarantees that the success probability of this cheating is negligible. Thus, $\overline{r} \approx \overline{\rho}$.

In a similar way, we can prove that this lemma holds even if there are at most $t-1$ malicious users. □

Theorem 1 (the distribution of the output). *In our framework, if the commitment scheme is perfectly hiding, the secret sharing scheme is a t-out-of-n VSS, and there are at most* $t-1$ *malicious users, then the probability ensemble defined by the order* $(o_1, o_2, \ldots, o_n)$ *and the probability ensemble defined by a permutation uniformly chosen over* $\{1, 2, 3, \ldots, n\}$ *are computationally indistinguishable.*

Proof. Let $\overline{o}$ denote the probability ensemble defined by the order $(o_1, o_2, \ldots, o_n)$ and $\overline{\sigma'}$ denote the probability ensemble defined by a permutation uniformly chosen over $\{1, 2, 3, \ldots, n\}$. Then we want to show $\overline{o} \approx \overline{o'}$.

Let us focus on step 3 in phase 2. We know that $\overline{o}$ is a function of $\overline{\rho}$, denoted by $\overline{o} = f(\overline{\rho})$. Following Lemma 2, we have $\overline{r} \approx \overline{\rho}$. Thus, $f(\overline{\rho}) \approx f(\overline{r})$. Furthermore, we have $\overline{o} \approx f(\overline{r})$. Note that $f(\overline{r})$ describes the case when all users are honest. From Lemma 1, we have $\overline{o'} \approx f(\overline{r})$. Finally, we obtain $\overline{o'} \approx \overline{o}$. □

Remark 1. The distribution of the order obtained by our protocol is not uniformly distributed. Note that $\overline{o} = f(\overline{\rho})$ and the string ρ is not uniformly distributed. From the proof of Lemma 2, we know that the malicious user may break the binding in step 1, phase 2, although the probability is negligible. One may prefer to employ a perfectly binding commitment scheme instead. In this case, the malicious users may break the computational hiding of the scheme in step 1, phase 1, although the probability is negligible too, and bias the distribution of ρ.

Remark 2. It is easy to verify that Theorem 1 still holds even if the commitment scheme is computationally binding and computationally hiding.

Theorem 2 (undeniability). *In our framework, if the commitment scheme is perfectly hiding, the secret sharing scheme is a t-out-of-n VSS, and there are at most $t-1$ malicious users; then no user can deny his secret after revealing his commitment of the secret to others.*

Proof. Note that the threshold of *t-out-of-n* VSS is set to be $t = [n/2]$, to generate subshares for users. Thus, after making any commitment of secret, the user can no longer deny the commitment. If the user tries to deny the commitment by either releasing a fake subsecret or releasing no information, the subsecret can still be recovered by honest users. □

Theorem 3 (secrecy). *In our framework, if the commitment scheme is perfectly hiding, the secret sharing scheme is a t-out-of-n VSS, and there are at most $t-1$ malicious users; then each subsecret of the user cannot be recovered from its commitment and is protected by a threshold SS.*

Proof. This property should be satisfied using a *t-out-of-n* VSS with a PHC commitment scheme as defined by Definition 2. □

4.4. Efficiency. In phase 1, each user needs to employ a one-time share sharing algorithm G to generate subshares for other users, a one-time commitment algorithm PHC to commit his secret and $(n-1)$-time share verification algorithms V to verify subshares received from other users. In phase 2, if we assume that all users act honestly by releasing their secrets, each user needs to employ $(n-1)$-time commitment verification algorithm to verify each released secret of other users. Thus, we can conclude that the complexity of using this approach is $O(n)$ for a group with n members.

To the best of our knowledge, our proposed scheme is the first MDS that can be used for a group of players to fairly determine the order of the group in applications. In particular, the coin-flipping protocol is a special type of MDS, that is, there are only two players. Although multiparty coin-flipping protocols [19–21] have been developed recently, these protocols are restricted for multiple parties to jointly choose one of the two possible outputs, which are different from our MDS.

When we compare our protocol with the coin-flipping protocol, our protocol is more efficient. As we have mentioned earlier in the introduction that employing a coin-flipping protocol repeatedly can achieve the same objective as ours. However, the coin-flipping protocol works two devices at a time to determine their order. That is, we can use the coin-flipping protocol in a straightforward manner to provide a solution for a general MDS. In this approach, all players are arranged on the leaves of a binary tree. Using a coin-flipping protocol between two users can determine a "winner." Repeatedly executing the coin-flipping protocol multiple times following the tree structure (i.e., complexity $O(n)$ can determine the "1st winner" among n users). Then, using the same approach on the remaining $n-1$ users can determine the "2nd winner" and so on. However, this approach is not effective because the complexity of using this approach is $O(n^2)$ for a group with n members. It is a time-consuming process. Thus, for large size of devices, it takes too much computational delay to determine their final order. In our proposed protocol, all devices work together at once to determine their order with the complexity $O(n)$. We are currently building a test platform. With the completion of this platform, we will take practical measurements and make comparisons with other approaches as described in [43] for our future work.

5. Concrete Instantiation

5.1. Protocol. We illustrate the detail of the proposed protocol by a concrete instantiation in Figure 4, where we use Pedersen's verifiable secret sharing (VSS) to allow each user to commit to a secret in the first phase. As follows, the user uses Shamir's secret sharing to divide the secret into multiple shares and share it among other users. The validity of these shares can be verified using VSS. In the second phase, each user releases his secret to the other users. If the user tries to deny his commitment by either releasing a fake secret or refusing to release the secret, misbehavior can be detected. At this moment, the other honest users can work together to recover this secret. This feature, called undeniability, can prevent the users from denying their secrets after making the commitments. The output of MDS $(o_1, o_2, \ldots, o_n)$ is a function of all secrets $(s_1, s_2, \ldots, s_n)$ of users.

5.2. Efficiency and Features. In this section, we evaluate the efficiency and summarize the features of this concrete protocol.

Efficiency. In our proposed protocol, all users work together to determine the output. Our protocol does not depend on any mutually trusted party. At the beginning of

Phase 1: Commitment phase

There are some public parameters, where p is a prime and $g, h \in Z_p$, and assumes that no one knows $\log_g h$.

Step1. Each user, U_i, randomly selects a sub-secret, s_i, and computes and publishes a commitment $\sigma(s_i, k_i) = g^{s_i} h^{k_i} = \sigma_0^i \bmod p$ where k_i is a random integer with $k_i \in Z_p$.

Step2. Each user, U_i, chooses two random polynomials, $f_i(x) = s_i + a_{i,1}x + \cdots + a_{i,t-1}x^{t-1} \bmod p$ and $g_i(x) = k_i + b_{i,1}x + \cdots + b_{i,t-1}x^{t-1} \bmod p$, with degree $t-1$ and $f_i(0) = s_i$ and $g_i(0) = k_i$. U_i generates a pair of sub-shares, $(f_i(x_j), g_i(x_j))$, for each other user, $j = 1, 2, \cdots, n$, $j \neq i$. Each pair of sub-shares, $(f_i(x_j), g_i(x_j))$ are sent to user, U_j, secretly.

Step 3. U_i computes and publishes commitments to the coefficients of the polynomials, $f_i(x)$ and $g_i(x)$ as $\sigma(a_{i,j}, b_{i,j}) = g^{a_{i,j}} h^{b_{i,j}} \bmod p = \sigma_j^i$, $j = 1, 2, \ldots, t-1$.

Step 4. Each pair of sub-shares, $(f_j(x_i), g_j(x_i))$ received from other user, U_j, can be verified by user, U_i, by checking whether $\sigma(f_j(x_i), g_j(x_i)) \stackrel{?}{=} \prod_{l=0}^{t-1} \sigma_l^{j^l} \bmod p$. If it passes the verification, the pair of sub-shares are verified; otherwise, user, U_i, requests user, U_j, to re-generates a new pair of sub-shares.

Phase 2: Open-commitment phase

Step1. Each user, U_i, releases his pair of sub-secrets, (s_i, k_i), to all other users.

Step 2. Each pair of sub-secrets, (s_i, k_i), of user, U_i, can be verified by checking whether $\sigma_0^i \stackrel{?}{=} g^{s_i} h^{k_i} \bmod p$. If the sub-secrets cannot be verified successfully, all other users should try to recover the sub-secret, s_i, using their sub-shares, $f_i(x_j)$ $j = 1, 2, \ldots, n$, $j \neq i$. Note that each sub-share $f_i(x_j)$ received from user, U_i, can also be verified following Step 4, in Phase 1. For example, if sub-shares, $f_i(x_j)$, $j = 1, 2, \cdots, l$, $l \geq t$ have been verified, the sub-secret, S_i, can be computed using the Lagrange interpolation formula as $s_i = \sum_{j=1}^{l} f_i(x_j) \prod_{k=1, k\neq j}^{l} \frac{-x_k}{x_j - x_k} \bmod p$.

Step 3. After obtaining all sub-secrets, s_i, $i = 1, 2, \ldots, n$ each user computes $\rho \leftarrow \bigoplus_{j=1}^{n} s_j$ and evenly divides ρ into n strings of bit sequences, $\rho_1, \rho_2, \ldots, \rho_n$. U_i sorts these n strings in a descending order. Let ρ_i denote the o_i-th smallest string of bit sequences in the set of strings. Then U_i obtains his order o_i and the orders of other users. Finally, U_i outputs the order $(o_1, o_2, \ldots, o_n)$.

Figure 4: Concrete instantiation.

each phase, each user needs to compute and release values to others. But there has no interaction among users to verify shares and to determine the output. In the first phase, each user needs to execute $2t$ modular exponentiations to compute his commitments and $2(t+1)$ modular exponentiations to verify each pair of subshares. Overall, each user needs $2(n-1)(t+1)$ modular exponentiations to verify all subshares from other users. In the second phase, each user needs two modular exponentiations to verify each subsecret of other users. Overall, each user needs $2(n-1)$ modular exponentiations to verify all subsecrets from other users.

In summary, we list the features of our proposed protocol as follows:

(1) The output of the protocol depends on inputs generated by all users and is uniformly distributed.

(2) Subshares generated by the owner of the subsecret can be verified by other users using the commitments of the subsecret.

(3) Once subshares are successfully verified in the first phase, the user can no longer deny his subsecret of the commitment. If the user tries to deny his subsecret, the subsecret can be recovered by honest users

and is used in the evaluation of the output in the second phase.

(4) The protocol can resist colluded users working together to attack the security of our protocol.

6. Conclusion

We propose a novel cryptographic primitive, called multiparty undeniable drawing straws (MUDS), that allows n users to work together to determine the order of users. MUDS is very useful in multiparty applications since it provides a fair way of determining an order of users. MUDS can also prevent cheaters from taking advantage of honest users by releasing their values last. Our proposal is more efficient and secure than the state-of-the-art cryptographic solutions, so it is absolutely attractive for multiparty applications in RIS towards 5G.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grants nos. 61772224, 62172181, and 62072133), the National Natural Science Foundation of China (Grants nos. U21A20465, 61922045, and U1836115), and the key projects of Guangxi Natural Science Foundation (no. 2018GXNSFDA281040).

References

[1] G. Hernandez, O. Arias, D. Buentello, and Y. Jin, *Smart Nest Thermostat—A Smart Spy in Your home*University of Central Florida, Orlando, FL, USA, 2014.

[2] A. G. Illera and J. V. Vidal, *Lights off! the Darkness of the Smart Meters,* BlackHat, Europe, 2014.

[3] M. Kabay, *Attacks on Power Systems: Hackers*, Malware, Santa Clara, CA, USA, 2010.

[4] K. Koscher, A. Czeskis, F. Roesner et al., "Experimental security analysis of a modern automobile," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2010.

[5] B. Miller and D. Rowe, "A survey SCADA of and critical infrastructure incidents," in *Proceedings of the Research in Information Technology (RIIT)*, Calgary, Alberta, Canada, October 2012.

[6] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," in *Whitepaper,* Black Hat, USA, 2014.

[7] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.

[8] J. Vijayan, "Stuxnet renews power grid security concerns," *Computerworld*, vol. 26, 2010.

[9] J. Xiong, R. Bi, Y. Tian, X. Liu, and D. Wu, "Toward lightweight, privacy-preserving cooperative object classification for connected autonomous vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2787–2801, 2022.

[10] D. M. Nicol, "Hacking the lights out," *Scientificfic American*, vol. 305, no. 1, pp. 70–75, 2011.

[11] A. Soullie, "Industrial control systems: pentesting PLCs 101," in *Whitepaper,* Black Hat, Europe, 2014.

[12] A. R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and Privacy Challenges in Industrial Internet of Things," in *Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, June 2015.

[13] M. Blum, "Coin flipping by telephone a protocol for solving impossible problems," *ACM SIGACT News*, vol. 15, no. 1, pp. 23–27, 1983.

[14] D. Aharonov, A. Ta-Shma, U. V. Vazirani, and A. C. Yao, "Quantum bit escrow," in *Proceedings of the STOC'00: Thirty-Second Annual ACM Symposium on Theory of Computing*, pp. 705–714, New York, NY, USA, May 2000.

[15] A. Ambainis, "A new protocol and lower bounds for quantum coin flipping," *Thirtieth Annual ACM Symposium on Theory of Computing*, vol. 68, no. 2, pp. 398–416, 2004.

[16] R. W. Spekkens and T. Rudolph, "Degrees of concealment and bindingness in quantum bit commitment protocols," *Physical Review A*, vol. 65, Article ID 012310, 2001.

[17] A. Nayak and P. Shor, "Bit-commitment-based quantum coin flipping," *Physical Review A*, vol. 67, no. 1, Article ID 012304, 2003.

[18] T. Moran, M. Naor, and G. Segev, "An optimally fair coin toss," *Theory of Cryptography Lecture Notes in Computer Science*, vol. 5444, pp. 1–18, 2009.

[19] A. Beimel, E. Omri, and I. Orlov, "Protocols for multiparty coin toss with dishonest majority," *Advances in Cryptology*, vol. 6223, pp. 538–557, 2010.

[20] I. Haitner and E. Tsfadia, "An almost-optimally fair three-party coin-flipping protocol," *SIAM Journal on Computing*, vol. 46, no. 2, pp. 408–416, 2014.

[21] A. Ambainis, H. Buhrman, Y. Dodis, and H. Röhrig, "Multiparty quantum coin flipping," in *Proceedings of the 19th IEEE Annual Conference on Computational Complexity*, pp. 250–259, Amherst, MA, USA, June2004.

[22] W. Liu, S.-S. Luo, and P. Chen, "A study of secure multi-party ranking problem," in *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence Networking and Parallel/Distributed Computing*, pp. 727–732, Qingdao, China, August 2007.

[23] A. C. Yao, "How to generate and exchange secrets," in *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pp. 218–229, Toronto, Canada, October 1986.

[24] C. Cheng, Y.-L. Luo, C.-X. Chen, and X.-K. Zhao, "Research on Secure Multi-Party Ranking Problem and Secure Selection Problem," in *Proceedings of the International Conference on Web Information Systems and Mining (WISM)*, Sanya, China, October 2010.

[25] J. Halpern and V. Teague, "Rational secret sharing and multiparty computation: extended abstract," in *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing - STOC'04*, pp. 623–632, New York, NY, USA, June 2004.

[26] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[27] G. Fuchsbauer, J. Katz, and D. Naccache, "Efficient Rational Secret Sharing in Standard Communication Networksficient

rational secret sharing in standard communication networks," in *Proceedings of the 7th Theory Of Cryptography Conference-TCC'10, LNCS 5978*, pp. 419–436, Berlin, Germany, February 2010.

[28] S. J. Ong, D. C. Parkes, A. Rosen, and S. P. Vadhan, "Fairness with an honest minority and a rational majority," in *Proceedings of the 6th Theory of Cryptography Conference-TCC'09, LNCS 5444*, pp. 419–436, San Francisco, CA, USA, March 2009.

[29] C. Tartary, H. Wang, and Y. Zhang, "An efficient and information theoretically secure rational secret sharing scheme based on symmetric bivariate polynomials," *International Journal of Foundations of Computer Science*, vol. 22, no. 6, pp. 1395–1416, 2011.

[30] M. Tompa and H. Woll, "How to share a secret with cheaters," *Journal of Cryptology*, vol. 1, no. 3, pp. 133–138, 1988.

[31] W. K. Moses Jr., and C. Pandu Rangan, "Rational secret sharing over an asynchronous broadcast channel with information theoretic security," *International Journal of Network Security & its Applications*, vol. 3, no. 6, pp. 1–18, 2011.

[32] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proceedings of the 6th IEEE Symposium on Foundations of Computer Science*, pp. 427–437, Los Angeles, CA, USA, October 1987.

[33] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology—CRYPTO'91,* Springer-Verlag, Berlin, Germany, 1992.

[34] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults," in *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (SFCS 1985)*, pp. 383–395, Portland, OR, USA, October 1985.

[35] J. C. Benaloh, "Secret sharing homomorphisms: keeping shares of a secret," *Advances in Cryptology*, vol. 263, pp. 251–260, 1987.

[36] M. Stadler, "Publicly verifiable secret sharing," *Advances in Cryptology*, vol. 3, pp. 190–199, 1996.

[37] B. Schoenmakers, "A simple publicly verifiable secret sharing scheme and its application to electronic voting," in *Proceedings of the Advances in Cryptology-CRYPTO'99, LNCS 1666*, pp. 148–164, Santa Barbara, CA, USA, August 1999.

[38] A. Peng and L. Wang, "One publicly verifiable secret sharing scheme based on linear code," in *Proceedings of the 2nd Conference on Environmental Science and Information Application Technology*, pp. 260–262, Wuhan, China, July 2010.

[39] A. Ruiz and J. L. Villar, "Publicly verifiable secret sharing from Paillier's cryptosystem," in *Proceedings of the WEWoRC'05, LNI P-74*, pp. 98–108, Leuven, Belgium, January2005.

[40] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the Advances in Cryptology-EUROCRYPT'99, LNCS 1592*, pp. 223–238, Prague, Czech Republic, May 1999.

[41] Y. Tian, C. Peng, and J. Ma, "Publicly verifiable secret sharing schemes using Bilinear pairings," *International Journal on Network Security*, vol. 14, no. 3, pp. 142–148, 2012.

[42] T.-Y. Wu and Y.-M. Tseng, "A pairing-based publicly verifiable secret sharing scheme," *Journal of Systems Science and Complexity*, vol. 24, no. 1, pp. 186–194, 2011.

[43] D. Wang, W. Li, and P. Wang, "Measuring two-factor authentication schemes for real-time data access in industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4081–4092, 2018.