The ISC Int'l Journal of Information Security

January 2018, Volume 10, Number 1 (pp. 63–69)

http://www.isecure-journal.org

Short Paper

# A Decentralized Online Sortition Protocol

Rasoul Ramezanian [1,*], and Mohsen Pourpouneh [2]

[1] *Ferdowsi University of Mashhad, Department of Mathematical Sciences, Mashhad, Iran*

[2] *Sharif University of Technology, Department of Mathematical Sciences, Tehran, Iran*

**ARTICLE INFO.**

*Article history:*

**Received:** 1 November 2017

**First Revised:** 29 December 2017

**Last Revised:** 15 January 2018

**Accepted:** 25 January 2018

**Published Online:** 31 January 2018

*Keywords:*

Sortition, Protocols, Mechanism Design.

**Abstract**

Sortition is widely used in social and economic mechanisms whenever it is required to choose one of the participants at random in a fair manner. On the other hand, by growth of the Internet, social and economic mechanisms are required to perform online sortition procedures in such a way that users can trust them. We propose a novel online sortition protocol, which is decentralized, and the winner of the sortition is chosen with the aid of all participants. In the real life, the sortition takes place via the lottery machine, which satisfies fairness, randomness, non-repudiation and openness. We argue that our proposed protocol also satisfies all these properties.



## 1 Introduction

Sortition is the process of making random choices and it is used in competitions (such as determining which teams must play against each other in a tournament), markets (such as determining the winner of a lottery), and economic mechanism (such as those used in random probability mechanism [1, 2]). In real world, the sortition could simply be done using a lottery machine (Figure 1). As this process can be done in front of all participants, almost every participants can be assured about the fairness, randomness, non-repudiation, and openness properties of the process. However, as the number of participants increases, using a lottery machine becomes almost impossible.

There are many applications for sortition protocols. One of them is the lottery procedure that the banks use in order to determine the winner. Also, there are many online websites that require to use the sortition procedure. Sortition protocols are also widely used in market and economic mechanisms [3–6] as randomized mechanisms. In a general mechanism design problem, the designer aims to satisfy three notions of truth-telling, fairness, and efficiency. To satisfy fairness, almost all mechanisms use lottery and online sortition protocols is a necessary tools to apply fair mechanisms in electronic markets.

An immediate solution is to use computers (i.e., a centralized approach) for generating random numbers, like using a pseudo random number generator. The problem with this solution is that the participants might not believe in the fairness and correctness of the algorithm that is used for generating random numbers. Also, it might be the case that the code is algorithm is coded in such a way that it always generates numbers within an specific interval, so that always participant with specific numbers are announced as the winners.

Decentralization is the process of distributing powers, credits, decision, etc. without using a central authority. As a method in designing of protocols, decen-

* Corresponding author.

Email addresses: rramezanian@um.ac.ir(R. Ramezanian), mohsen.pourpoune1@student.sharif.ir(M. Pourpouneh)

ISSN: 2008-2045 

**Figure 1**. A lottery machine

tralization is becoming more and more popular in the world. This is a new method for constructing scalable and multi agent applications. Block chain is a famous example of a decentralized protocol and cryptocurrencies is one of the applications of block chain. Decentralized protocols [1] are more flexible, transparent, distributed, and resilient.

In this paper, we propose a sortition protocol which allows one to determine the winner in a decentralized manner. That is the winner is chosen in such a way that "nobody and everybody" chooses him. In words, there exists no specific participant who chooses the winner and all the participants contribute in determining the winner. We propose a protocol which can carry out a secure sortition over the world wide web in such a way that it satisfies several desired properties. The protocol is such that the winner is chosen with the aid of all participants, and all the participants can check the fairness and true randomness of the output. Moreover, all the participants are assured about the non-repudiation, and openness properties of the protocol.

The main idea of the proposed decentralized sortition protocol for two participants $A_0$ and $A_1$ is as follows: First, each participant secretly generates a random bit. Let $b_0$ and $b_1$ denote the random bit generated by $A_0$ and $A_1$, respectively. Set $b = b_0 \oplus b_1$. Now, if $b = 0$, then $A_0$ is announced as the winner, and if $b = 1$ then $A_1$ is announced as the winner. In this way, the winner is chosen in a decentralized manner.

## 2 Related Work

In the literature, there are some few sortition protocols which tries to satisfy the desired properties of the real world sortition process.

One of the most well known protocols in this regard, is the *secure coin flipping* protocol initially suggested by Manuel Blum [10]. The main concern in a coin tossing protocol is the prevention of generating a bias output. In this protocol, two agents are about to toss a coin over the phone in such a way that it is fair. The proposed protocol guarantees that both agents will pick their sequence of bits at random, and that non of them knows what sequence of bits the other one is using. The idea of coin-tossing can be extended to multiparty protocols. In [11], it is shown that if more than half of the parties are honest, then we can have protocols with negligible bias. Also, in [12] it is shown that when at least half of the parties might be malicious then every protocol of $r$ round will have a bias of $\Omega(1/r)$.

An electronic sortition protocol is proposed in [13] where they aimed to find a fair order in a group. Their protocol satisfies openness, randomness, and fairness properties. In order to achieve these properties, they use digital signature, public key cryptography, and a disturbance technique. In comparison, our protocol is less complicated, and also it is decentralized.

In [14] the authors argue that the security requirements of online lotteries are similar to those of online voting. They develop a sortition protocol which does not rely on a trusted third party, and the random process is distributed to all players using the protocol.

In [15] a lottery protocol is introduced. They introduce a denial of service attack to the protocol. The attack takes place if an agent does not reveal his secret message. In our revised protocol (Section 5) we came up with a solution to prevent this issue. Also, the work in [16] is based on a delaying function, which prevents computationally-bounded adversaries from cheating. Although, their proposed lottery scheme is vulnerable to the denial of service attack.

In [17] the authors propose an E-lottery scheme, based on verifiable random functions [18]. The protocol does not rely on a trusted third party. Each participant sends a ticket to the dealer and then he links all the tickets with a hash function. The result generation phase, uses a verifiable random function. In the final step the winning player reveals his keys and claims his prize.

## 3 Properties of the sortition protocol

There are several properties that a real life sortition procedure satisfies. In this section we first recall some of these properties. Later, we will prove that our online protocol also satisfies these properties.

- **Fairness**: A sortition protocol satisfies fairness if an only if all the participants have access to the same information. In the real life sortition process, since all participants are present and see the actual procedure, this property is simply satisfied.

[1] One can refer to [7–9] for more examples of the applications based on the decentralization solution.

- **Randomness**: The result of the sortition protocol is random and can not be predicted. Also, all the participants have equal chance to be chosen by the protocol. In the real life sortition procedure, it is believed by the nature rules that the chance of all the balls in the lottery machine is the same.
- **Openness**: The process of the sortition and the state of the system is open to all the participants. In the real life sortition process, the openness property is satisfied because the process takes place face to face.
- **Unforgeability**: Nobody can forge the actions of the participants and the final result of the system. In the real life sortition process, since participants believe the organizers, this property is satisfied.
- **Non-repudiation**: The participants can not deny their actions.
- **Verifiability**: All the participants can verify the result of the system. It is obvious that this property is satisfied in the real life sortition procedure.
- **Anti-collusion**: No one can collude with others to cheat in the sortition outcome. In the real life sortition process, this is due to the fact that the organizer is a trusted third party who checks that every participant writes his own name on the ball [2].

## 4 Protocol Specification

Our sortition protocol **P** consists of two phases: initialization and execution. In the initial phase we suppose that there is an sortition *Executer* [3] $E$, and a set of participants $P = \{P_1, P_2, \ldots, P_n\}$ such that $n = 2^k$ for some natural number $k$. We assume that each participant $i$ has a pair of public key/private key $(pk_i, sk_i)$. We also assume that bulletin board where the executer and all the participants can publicly broadcast their messages on it. There are five rounds in the execution phase.

- Round 1: The executer $E$, generates a session Identifier [4] $S^{ID}$ and assigns a unique $IS^i$, which is an string in $\{0,1\}^k$, to each participants $i$ as his identity number. The executer broadcasts $S^{ID}$ and $IS^i$ together with the names of each participant as the rows of the bulletin board (See Table 1).
- Round 2: Each participants $P_i$ generates a uniformly random string $w^i \in \{0,1\}^k$. Then he signs the triple $(w^i, S^{ID}, IS^i)$ and concatenates it with a time stamp $T$ and his name $P_i$. Finally, he encrypts the whole message using his public key and sends the message $R_i^2 := \{\{(w^i, S^{ID}, IS^i)\}_{sk_i}, T, P_i\}_{pk_i}$ to the bulletin board (See Table 2).
- Round 3: Each participant decrypts its message in the second round (and publishes all the necessary information for the the verification of its correctness such as the value of the used randomness), and broadcasts it in front of its name in the bulletin board. Since everyone has access to all the public keys, each participant can verify that if the messages published at round 2 are exactly the same as those encrypted in the third round (See Table 3).
- Round 4: Since everybody has access to all the public keys each participant can obtain $w^i$s. On the fourth round, the executer broadcasts all the $w^i$s in front of the name of each participant. (see Table 4).
- Round 5: The executer, applies the XOR function on all the lottery strings $\{w^i\}$. Then, it broadcast it in the public table (See Table 5).

$$w = w^1 \oplus w^2 \oplus \cdots \oplus w^n$$

Finally, the winner of the sortition is the participant whose identity number is the same as $w$. That is, the winner is the participant with $IS^j = w$.

There are a few remarks about the protocol that ought to made.

- We can slightly modify the protocol such that the role of the executer is less demanding. To do so, each participant $i$ can refer to the lottery website and suggest an $IS^i$ for himself. The $IS^i$ is assigned to him if it is a valid one (i.e., if $IS^i \in \{0,1\}^k$), and it is not assigned to someone else so far.
- The timestamp $T$ is used for those cases in which the lottery has a time limit for participation. That is, we might only use those strings that are generated within the last hour, or days.
- In the second round we require each agent to encrypt his message and decrypt it later in the third round. The reason is that we want all the participants to generate and publish their lottery strings all at the same time (in the situation that no participant is aware of the lottery strings of others when he is generating his own lottery string).
- In our protocol we require each agent to be assigned an $IS^i$. The reason is that $P_i$ is the ac-

[2] The agents might have incentive to write the same name on the paper so that they have higher chance of winning.
[3] The executer can be regarded as the owner of the website which is about to perform to sortition for some purpose.
[4] The session identifier is used in case when there are several parallel sessions, and to identify that all the random strings belong to the ongoing sortition.

**Table 1**. First round of the execution part

| | Round 1: Broadcast by the executer | Round 2: | Round 3: | Round 4: | Round 5: |
|---|---|---|---|---|---|
| $E$ | $S^{ID}$ | – | – | – | |
| $P_1$ | $IS^1$ | – | – | – | – |
| $P_2$ | $IS^2$ | – | – | – | – |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $P_n$ | $IS^n$ | – | – | – | – |

**Table 2**. Second round of the execution part

| | Round 1: Broadcast by the executer | Round 2: Broadcast by each participant | Round 3: | Round 4: | Round 5: |
|---|---|---|---|---|---|
| $E$ | $S^{ID}$ | – | – | – | – |
| $P_1$ | $IS^1$ | $\{\{(w^1, S^{ID}, IS^1)\}_{sk_1}, T, P_1\}_{pk_1}$ | – | – | – |
| $P_2$ | $IS^2$ | $\{\{(w^2, S^{ID}, IS^2)\}_{sk_2}, T, P_2\}_{pk_2}$ | – | – | – |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $P_n$ | $IS^n$ | $\{\{(w^n, S^{ID}, IS^n)\}_{sk_n}, T, P_n\}_{pk_n}$ | – | – | – |

**Table 3**. Third round of the execution part

| | Round 1: Broadcast by the executer | Round 2: Broadcast by each participant | Round 3: Broadcast by each participant | Round 4: | Round 5: |
|---|---|---|---|---|---|
| $E$ | $S^{ID}$ | – | – | – | – |
| $P_1$ | $IS^1$ | $R_1^2$ | $\{(w^1, S^{ID}, IS^1)\}_{sk_1}, T, P_1$ | – | – |
| $P_2$ | $IS^2$ | $R_2^2$ | $\{(w^2, S^{ID}, IS^2)\}_{sk_2}, T, P_2$ | – | – |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $P_n$ | $IS^n$ | $R_n^2$ | $\{(w^n, S^{ID}, IS^n)\}_{sk_n}, T, P_n$ | – | – |

**Table 4**. Fourth round of the execution part

| | Round 1: Broadcast by the executer | Round 2: Broadcast by each participant | Round 3: Broadcast by each participant | Round 4: Broadcast by the executer | Round 5: |
|---|---|---|---|---|---|
| $E$ | $S^{ID}$ | – | – | – | – |
| $P_1$ | $IS^1$ | $R_1^2$ | $\{(w^1, S^{ID}, IS^1)\}_{sk_1}, T, P_1$ | $w^1$ | – |
| $P_2$ | $IS^2$ | $R_2^2$ | $\{(w^2, S^{ID}, IS^2)\}_{sk_2}, T, P_2$ | $w^2$ | – |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $P_n$ | $IS^n$ | $R_n^2$ | $\{(w^n, S^{ID}, IS^n)\}_{sk_n}, T, P_n$ | $w^n$ | – |

**Table 5**. Fifth round of the execution part

| | Round 1: Broadcast by the executer | Round 2: Broadcast by each participant | Round 3: Broadcast by each participant | Round 4: Broadcast by the executer | Round 5: Broadcast by the executer |
|---|---|---|---|---|---|
| $E$ | $S^{ID}$ | – | – | – | $w$ |
| $P_1$ | $IS^1$ | $R_1^2$ | $\{(w^1, S^{ID}, IS^1)\}_{sk_1}, T, P_1$ | $w^1$ | $w$ |
| $P_2$ | $IS^2$ | $R_2^2$ | $\{(w^2, S^{ID}, IS^2)\}_{sk_2}, T, P_2$ | $w^2$ | $w$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $P_n$ | $IS^n$ | $R_n^2$ | $\{(w^n, S^{ID}, IS^n)\}_{sk_n}, T, P_n$ | $w^n$ | $w$ |

tual identity [5] of the participant whereas $IS^i$ is a number in the range of 0 to $2^k$, which maps the agents to their identity. Also, as each agent generates a string in $\{0,1\}^k$ this will guarantee that the output of the sortition protocol is definitely among the participants.
- In this protocol we used signature and encryption schemes, so that we can keep the protocol simple and use the most basic schemes of cryptography. However, it should be noted that the one can use commitment schemes instead of signature and encryption scheme in our proposed protocol.

**Theorem 1.** *The sortition protocol* **P** *satisfies fairness, randomness, openness, unforgeability, non-repudiation, verifiability and anti-collusion properties.*

*Proof.* We discuss that the protocol satisfies these properties.

- Fairness: It is easily seen that all the participants have access to equal information. In every round the bulletin board is publicly accessible, and everyone knows his private key. Also all the agents learn the lottery strings in round 4. Hence, all the agents have access to all the necessary information.
- Randomness: Since each $w^i$ is randomly selected and $w$ is the XOR [6] of all the $w^i$s, the protocol satisfies randomness property.
- Openness: It is trivial.
- Unforgeability: On the second round, each participant $P_i$ freely can generate a random string $w^i$, and signs $\{w^i, S^{ID}, IS^i\}$.

$$R_i^2 = \{\{(w^i, S^{ID}, IS^i)\}_{sk_i}, T, P_i\}_{pk_i}$$

Because of the signature, we are certain that the string $w^i$ is generated by the participant $P_i$. Also, due to the time stamp $T$ we are assured that the random number is fresh. Furthermore, due to the session ID, $S^{ID}$, the participants are assured that the lottery string belongs to the current session.
- Non-repudiation: Since the message $\{w^i, S^{ID}, IS^i\}_{sk_i}$, is signed by the participant $P_i$ and every participants have access to public keys they can verify that the messages in the third round are the decryption of the messages in the second round. Therefore, no one can deny his the random string $w^i$.
- Verifiability: In the third round all the agents can verify the true decryption of messages in the second round. Also, by use of public keys they can find all the $w^i$s, and verify the outcome.
- Anti-Collusion: Suppose $A$ is the set of all participants and the members of set $I \subsetneq A$ collude with each other. (Since everyone knows all the executer's information we can omit $S$ from the collusion party.) Let $J = A \setminus I$, $w_I = \bigoplus\{w_i | P_i \in I\}$ and $w_J = \bigoplus\{w_j | P_j \in J\}$. Since $w_J$ is a random string in $\{0,1\}^k$ then $w_I \oplus w_J$ is also random. This proves the anti-collusion property.

□

## 5 The Revised Protocol

In Section 4 we assumed the number of participant to be a power of 2, which restricts the use case of our protocol. In this section we propose a revised version of our protocol in such a way that works with any arbitrary number of participants.

Suppose that $m$ is the number of participants and $2^{k-1} \leq m \leq 2^k$, for some natural number $k$. If run the protocol of Section 4, then the participants will generate random strings $w^i \in \{0,1\}^k$. Since $m$ is less than $2^k$ there might be cases in which there exists no $IS^j$ such that $IS^j = w^1 \oplus w^2 \oplus \cdots \oplus w^m$. To overcome this issue we propose a revised protocol $\mathcal{P}$. We slightly change the protocol **P** in the first, third and fifth rounds.

- Round 1: The executer generates a permutation set of strings $B = \{B(1), B(2), \ldots, B(2^k)\}$ where each $B(i)$ is a different string in $\{0,1\}^k$. The executer publishes the message $\{S^{ID}, \{\{S^{ID}, B\}_{sk_e}, T\}_{pk_e}\}$ on the bulletin board in first round where $S^{ID}$ is the session ID. Note that $(sk_e, pk_e)$ are the key pairs of the executer. The executer also generates the unique identity numbers, that is $IS^i \in \{0,1\}^k$ for each participant $i$ (See Table 6).
- Round 2: No change is required for this round.
- Round 3: The executer also decrypts its message in Round 1. Therefore the signed message $\{S^{ID}, B\}_{sk_e}$, is revealed in this round.
- Round 4: No change is required for this round.
- Round 5: Each participant (including the executer) can compute $w = w^1 \oplus w^2 \oplus \cdots \oplus w^m$. If there exists an $IS^j$, such that $IS^j = w$, then the participant $P_j$ is chosen as the winner. If not, the each participant decrypts the message $S^{ID}, \{\{S^{ID}, B\}_{sk_e}, T\}_{pk_e}$, and computes $c_1 = B(1) \oplus w$ as a potential winner. If there exists $IS^j$ such that $IS^j = c_1$, the participant $P_j$ is chosen as the winner, otherwise the string $c_2 = B(2) \oplus w$ must be computed, and if there exists $IS^j$ such that $IS^j = c_2$, the participant $P_j$ is

[5] Like his first name and his last and his social identity number.
[6] Note that the XOR operation preserves the uniform distribution.

**Table 6**. First round of the execution part when number of agents is arbitrary

| | Round 1: Broadcast by the executer | Round 2: | Round 3: | Round 4: | Round 5: |
|---|---|---|---|---|---|
| $E$ | $S^{ID}, \{\{S^{ID}, B\}_{sk_e}, T\}_{pk_e}$ | – | – | – | – |
| $P_1$ | $IS^1$ | – | – | – | – |
| $P_2$ | $IS^2$ | – | – | – | – |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $P_m$ | $IS^m$ | – | – | – | – |

**Table 7**. Third round of the execution part when number of agents is arbitrary

| | Round 1: Broadcast by the executer | Round 2: Broadcast by each participant | Round 3: Broadcast by each participant | Round 4: | Round 5: |
|---|---|---|---|---|---|
| $E$ | $S^{ID}, \{\{S^{ID}, B\}_{sk_e}, T\}_{pk_e}$ | – | $\{S^{ID}, B\}_{sk_e}$ | – | – |
| $P_1$ | $IS^1$ | $R_1^2$ | $\{(w^1, S^{ID}, IS^1)\}_{sk_1}, T, P_1$ | – | – |
| $P_2$ | $IS^2$ | $R_2^2$ | $\{(w^2, S^{ID}, IS^2)\}_{sk_2}, T, P_2$ | – | – |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $P_m$ | $IS^m$ | $R_m^2$ | $\{(w^m, S^{ID}, IS^m)\}_{sk_m}, T, P_m$ | – | – |

**Table 8**. Fifth round of the execution part when number of agents is arbitrary

| | Round 1: Broadcast by the executer | Round 2: Broadcast by each participant | Round 3: Broadcast by each participant | Round 4: Broadcast by the executer | Round 5: Broadcast by the executer |
|---|---|---|---|---|---|
| $E$ | $S^{ID}, \{\{S^{ID}, B\}_{sk_e}, T\}_{pk_e}$ | – | – | – | $w$ |
| $P_1$ | $IS^1$ | $R_1^2$ | $\{S^{ID}, B\}_{sk_e}$ | $w^1$ | $w$ |
| $P_2$ | $IS^2$ | $R_2^2$ | $\{(w^1, S^{ID}, IS^1)\}_{sk_1}, T, P_1$ | $w^2$ | $w$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $P_m$ | $IS^m$ | $R_m^2$ | $\{(w^m, S^{ID}, IS^m)\}_{sk_m}, T, P_m$ | $w^m$ | $w$ |

chosen winner. This process continues until one we can find some $c_i$ which corresponds to the identity string $IS^j$ for some participants $P_j$.

The output $w = w^1 \oplus w^2 \oplus \cdots \oplus w^m$ is a uniform distribution on $\{0,1\}^k$ and no participant can predict the final output of the protocol before the third round. Since $\{w \oplus B(i) \mid 0 \leq i \leq 2^k - 1\} = \{0,1\}^k$ and $w$ is uniformly distributed in $\{0,1\}^k$ we have each $c_i$ is uniformly distributed in $\{0,1\}^k$.

**Theorem 2.** *The revised sortition protocol satisfies fairness, randomness, openness, unforgeability, non-repudiation, verifiability and anti-collusion properties.*

*Proof.* The proof of fairness, openness, unforgeability, non-repudiation and anti-collusion are similar to the proof of the first version of the protocol. Hence, we only prove the following properties:

- Randomness: Since each $w^i$ is randomly selected by participant $i$, the values $\bigoplus w^i \oplus B(j)$ are random for $0 \leq j \leq 2^k$.
- Verifiability: In the third round all the participants can verify the true decryption of messages in the second round, and the executer message in the first round. Also, by use of public keys they can find all the $w^i$s, and determine the outcome.

□

In the third round of protocol **P**, if a participant refuses to decrypt his message in the second round, then the protocol will never terminate. Thus, **P** is vulnerable to denial of service attack. In the revised protocol $\mathcal{P}$, we can set a timeout for publishing message in each round and if a player refuses to send a message we simply will omit him from the protocol. This will not affect our procedure since the revised protocol $\mathcal{P}$ is applicable to any number of participants.

## 6 Conclusion

We proposed a decentralized sortition protocols and argued that our protocol satisfies fairness, randomness, non-repudiation and openness properties. In comparison to other several protocols that exist in the literature, our protocol is simpler to understand and execute. It should be noted that one can use modular summation instead of XOR, here we used XOR to keep the protocol as simple as possible.

Although the way that we recognize the winner

of the sortition is decentralized, but our protocol requires an executer to control the protocol. This is also natural in the sense that the executer can be considered as the bank owner or the website that is performing the sortition. For further works, we aim to propose a new decentralized sortition protocol where it is based on block chain and also the winner is chosen by contribution of all agents. To do this aim, we need to add a proof of work to manage messages of agents as blocks and prevents double sending attacks.

## References


[1] Atila Abdulkadiroğlu and Tayfun Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.

[2] Yeon-Koo Che and Fuhito Kojima. Asymptotic equivalence of probabilistic serial and random priority mechanisms. *Econometrica*, 78(5):1625–1672, 2010.

[3] Noam Nisan and Amir Ronen. Algorithmic mechanism design. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 129–140. ACM, 1999.

[4] Kenneth J Arrow, Amartya Sen, and Kotaro Suzumura. *Handbook of social choice and welfare*, volume 2. Elsevier, 2010.

[5] Felix Brandt, Vincent Conitzer, Ulle Endriss, Ariel D Procaccia, and Jérôme Lang. *Handbook of computational social choice*. Cambridge University Press, 2016.

[6] Atila Abdulkadiroglu and Tayfun Sönmez. School choice: A mechanism design approach. *The American Economic Review*, 93(3):729–747, 2003.

[7] Rafik Makhloufi, Grégory Bonnet, Guillaume Doyen, and Dominique Gaïti. Decentralized aggregation protocols in peer-to-peer networks: a survey. In *IEEE International Workshop on Modelling Autonomic Communications Environments*, pages 111–116. Springer, 2009.

[8] JA Alvarez Bermejo, MA Lodroman, and JA Lopez-Ramos. A decentralized protocol for mobile control access. *The Journal of Supercomputing*, 70(2):709–720, 2014.

[9] Airlie Chapman, Eric Schoof, and Mehran Mesbahi. Semi-autonomous networks: theory and decentralized protocols. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1958–1963. IEEE, 2010.

[10] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.

[11] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85. ACM, 1989.

[12] Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 364–369. ACM, 1986.

[13] Biao He and Yu Wei. Electronic sortition. In *The 2009 International Symposium on Intelligent Information Systems and Applications (IISA 2009)*, page 203, 2009.

[14] Stéphane Grumbach and Robert Riemann. Distributed random process for a large-scale peer-to-peer lottery. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 34–48. Springer, 2017.

[15] Arjen K Lenstra and Benjamin Wesolowski. A random zoo: sloth, unicorn, and trx. *IACR Cryptology ePrint Archive*, - -:366, 2015.

[16] David M Goldschlag and Stuart G Stubblebine. Publicly verifiable lotteries: Applications of delaying functions. In *International Conference on Financial Cryptography*, pages 214–226. Springer, 1998.

[17] Sherman SM Chow, Lucas CK Hui, Siu-Ming Yiu, and KP Chow. An e-lottery scheme using verifiable random function. In *International Conference on Computational Science and its Applications*, pages 651–660. Springer, 2005.

[18] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 120–130. IEEE, 1999.


**Mohsen Pourpouneh** was born in 1989 in Isfahan. He got his B.S. and M.S. in Computer Science, from Shahid Beheshti University and Tehran University, respectively. He started his career as a Ph.D. student at Sharif University of Technology. His research interest includes formal method, electronic voting protocols, and multi-agent systems.

**Rasoul Ramezanian** was born in Mashhad in 1979. He got his B.S. and M.S. in Mathematics. In 2008, he graduated from a Ph.D. program at Mathematical Sciences Department of Sharif University of Technology. He is an assistant professor at the faculty of Mathematical Sciences at Ferdowsi University of Mashhad. His research interests include formal method, specifying and verifying security protocols, multi-agent systems, and process algebra.