Brief Announcement: Statement Voting and Liquid Democracy

Bingsheng Zhang[*]
School of Computing and Communications
Lancaster University, UK
b.zhang2@lancaster.ac.uk

Hong-sheng Zhou
Department of Computer Science
Virginia Commonwealth University, USA
hszhou@vcu.edu

ABSTRACT

The existing (election) voting systems, e.g., representative democracy, have many limitations and often fail to serve the best interest of the people in collective decision making. To address this issue, the concept of liquid democracy has been emerging as an alternative decision making model to make better use of "the wisdom of crowds". Very recently, a few liquid democracy implementations, e.g. Google Votes and Decentralized Autonomous Organization (DAO), are released; however, those systems only focus on the functionality aspect, as no privacy/anonymity is considered.

In this work, we, for the first time, provide a rigorous study of liquid democracy under the Universal Composability (UC) framework. In the literature, liquid democracy was achieved via two separate stages – delegation and voting. We propose an efficient liquid democracy e-voting scheme that unifies these two stages. At the core of our design is a new voting concept called *statement voting*, which can be viewed as a natural extension of the conventional voting approaches. We remark that our statement voting can be extended to enable more complex voting and generic ledger-based non-interactive multi-party computation. We believe that the statement voting concept opens a door for constructing a new class of e-voting schemes.

CCS CONCEPTS

•**Security and privacy** → *Public key encryption; Mathematical foundations of cryptography; Distributed systems security;*

KEYWORDS

E-voting, UC, Liquid democracy, Statement Voting

1 INTRODUCTION

Elections/Referendums provide people in each society with the opportunity to express their opinions in the collective decision making process. The existing election/voting systems can be mainly divided into two types, direct democracy and representative democracy. Although the former one treats every voter equally, it is not scalable; therefore, the latter one is widely used in most countries. However, representative democracy has many limitations and it often fails to serve the best interest of the people. For example, to make correct decisions, the voters have to invest tremendous effort to analyze the issues. The cost of identifying the best voting strategy is high, even if we assume that the voter has collected accurate information. What's worse, misinformation campaigns often influence the voters to select certain candidates which could be against the voters' own interests. In the past decades, the concept of liquid democracy [6] has been emerging as an alternative decision making model to make better use of collective intelligence. Liquid democracy is a hybrid of direct democracy and representative democracy, where the voters can either vote directly on issues, or they can delegate their votes to representatives who vote on their behalf. Due to its advantages, liquid democracy has received high attentions since the spread of its concept; however, there is no satisfactory solution in the form of either paper-voting or e-voting yet[1]. Is it possible to introduce new technologies to circumvent the implementation barriers to liquid democracy?

2 A NEW CONCEPT

We could approach the above problem via multiple angles. In this paper, we propose a new and clean concept: *statement voting*. Statement voting can be viewed as a natural extension of traditional candidate voting. Instead of defining a fixed election candidate, each voter can define a statement in his or her ballot but leave the vote "undefined" during the voting phase. During the tally phase, the (conditional) actions expressed in the statement will be carried out to determine the final vote. Single Transferable Vote (STV) is a special case of statement voting, where the voters rank the election candidates instead of naming only one candidate in their ballots. The ranked candidate list together with the STV tally rule can be viewed as an outcome-dependent statement. Roughly speaking, the statement declares that if my favorite candidate has already won or has no chance to win, then I would like to vote for my second favorite candidate, and so on. In terms of liquid democracy, the vote delegation can be expressed as a target-dependent statement, where a voter can define that his/her ballot is the same as the target voter's ballot. Of course, the target voter can also state whether he/she is willing to be delegated in the ballot. To obtain the basic intuition, let's first leave privacy aside and consider the following toy example.

Example: Each ballot is in the form of $(\mathsf{ID}, \mathtt{action}, \mathsf{target})$. If a voter V_i is willing to be delegated, he/she sets $\mathsf{ID} := \mathsf{V}_i$; otherwise, sets $\mathsf{ID} := \perp$. If V_i wants to delegate his/her vote to another voter V_j, then the ballot is $B := (\mathsf{ID}, \mathtt{delegate}, \mathsf{V}_j)$. If V_i wants to directly

[*]This work is partially supported by EPSRC IoT Security Research Hub, EP/N023234/1.

PODC '17, Washington, DC, USA
© 2017 Copyright held by the owner/author(s). 978-1-4503-4992-5/17/07…$15.00
DOI: 10.1145/3087801.3087868

[1]All the existing liquid democracy implementations, e.g., Google Votes and Decentralized Autonomous Organization (DAO) do not consider privacy/anonymity. This drawback prevents them from being used in serious elections. Here, we note that straightforword blockchain-based solutions cannot provide good privacy in practice. Although some blockchains such as Zerocash [1] can be viewed as a global mixer, they implicitly require anonymous channels. While in practice, all the implementations of anonymous channels suffer from time leakage, i.e., the user's ID is only hidden among the other users who are also using the system at the same time. Subsequently, the adversary can easily identify the user during quiet hours.

vote for v_i, then the ballot is $B := (\mathsf{ID}, \mathtt{vote}, v_i)$. Suppose there are seven ballots: $B_1 := (\mathsf{V}_1, \mathtt{delegate}, \mathsf{V}_7)$, $B_2 := (\mathsf{V}_2, \mathtt{vote}, v_2)$, $B_3 := (\mathsf{V}_3, \mathtt{vote}, v_3)$, $B_4 := (\perp, \mathtt{vote}, v_4)$, $B_5 := (\mathsf{V}_5, \mathtt{delegate}, \mathsf{V}_4)$, $B_6 := (\perp, \mathtt{delegate}, \mathsf{V}_3)$ and
$B_7 := (\mathsf{V}_7, \mathtt{delegate}, \mathsf{V}_3)$. Here, the effective vote of B_1 is defined by B_7, which is further defined by B_3; note that B_3 votes for v_3; that means, B_7 votes for v_3 by following B_3. Now let's consider B_6: B_6 follows B_3; however, B_6 is not willing to be followed by anyone; as a result, B_6 also votes for v_3. Finally, let's consider B_5: B_5 follows B_4; however, B_4 is not willing to be followed by anyone; as a consequence, B_5 is re-defined as blank ballot, $\perp$. After interpreting the delegation statements, the final votes are $(v_3, v_2, v_3, v_4, \perp, v_3, v_3)$.

Careful readers may wonder why this type of natural voting idea has never appeared in the *physical* world. Indeed, it is typically not available in the real life. Different from the toy example, in the reality, the voters care about privacy and anonymity. To ensure anonymity, the voters are not willing to leave their identities in the ballots. If no identities (or equivalences) are included in the ballots, then it is difficult for voters to "follow" other voters' choices. The election committees might assign each voter a temporal ID to achieve anonymity, but a voter needs to obtain the target voter's temporal ID in order to delegate his vote. This requires secure peer-to-peer channels among all the voters, which is not practical. In "our design" paragraph in the Introduction, we will present the first *digital* construction for implementing full-fledged liquid democracy.

3 MODELING LIQUID DEMOCRACY VOTING

We for the first time provide a rigorous modeling for liquid democracy voting. More concretely, we model liquid democracy voting in the well-known Universal Composability (UC) framework, via an ideal functionality $\mathcal{F}_{\text{LIQUID}}$. The functionality interacts with a set of voters, trustees, and an auditor, and consists of preparation phase, voting/delegation phase, and tally phase. During the preparation phase, the trustees need to indicate their presence to $\mathcal{F}_{\text{LIQUID}}$, and the election/voting will not start until all the trustees have participated the preparation. During the voting/delegation phase, each voter can either directly vote for the candidate(s) or delegate his vote to another voter. In addition, each voter can use a flag α to indicate whether he is willing to be delegated. Note that, when all the trustees are corrupted, $\mathcal{F}_{\text{LIQUID}}$ leaks the voters' ballots to the adversary.

To model the privacy of mixing type of e-voting, we let $\mathcal{F}_{\text{LIQUID}}$ replace each voter's ID with a pseudonym. The functionality $\mathcal{F}_{\text{LIQUID}}$ then sorts the anonymized ballots lexicographically to hide the order correspondence and reveal them to the adversary. Note that this type of information leakage is consistent with the conventional paper-based voting system, where a voter submits his ballot to a ballot box and the ballot is mixed together with the other voters' ballots. To model end-to-end verifiability, we introduce an auditor Au who will verify whether the election result is mis-presented. Note that the auditor can be performed by any party, and here we model auditor as a single entity for simplicity.

We emphasize that in practice, virtually all threshold cryptographic systems suffer from selective failure attacks; namely, during the opening process of a threshold cryptographic system, the last several share holders can jointly see the content to be opened themselves before hand. Hence, they can decide if they want to

Functionality $\mathcal{F}_{\text{LIQUID}}$

The functionality $\mathcal{F}_{\text{LIQUID}}$ interacts with a set of voters $\mathbb{V} := \{\mathsf{V}_1, \ldots, \mathsf{V}_n\}$, a set of trustees $\mathbb{T} := \{\mathsf{T}_1, \ldots, \mathsf{T}_k\}$, an auditor Au, and the adversary $\mathcal{S}$. Let $\mathbb{V}_{\mathsf{honest}}$, $\mathbb{V}_{\mathsf{corrupt}}$ and $\mathbb{T}_{\mathsf{honest}}$, $\mathbb{T}_{\mathsf{corrupt}}$ denote the set of honest/corrupt voters and trustees, respectively. $\mathcal{F}_{\text{LIQUID}}$ is parameterized by an algorithm TallyProcess, a table Tab, and variables *result*, T_1, T_2, T_3, and B_i for all $i \in [n]$.

Initially, set *result* $:= \emptyset$, $T_1 := \emptyset$, $T_2 := \emptyset$, $T_3 := \emptyset$; for $i \in [n]$, set $B_i := \emptyset$.

Preparation:

(1) Upon receiving input (InitialTrustee, sid) from the trustee $\mathsf{T}_j \in \mathbb{T}$, set $T_1 := T_1 \cup \{\mathsf{T}_j\}$, and send a notification message (InitialTrusteeNotify, sid, T_j) to the adversary $\mathcal{S}$.

(2) Upon receiving input (InitialVoter, sid, η) from the voter $\mathsf{V}_i \in \mathbb{V}$, if $|T_1| < k$, ignore the input.
Otherwise, send (InitialVoterNotify, sid, V_i) to the adversary $\mathcal{S}$. If $|\mathbb{T}_{\mathsf{corrupt}}| = k$, then additionally send a message (DelLeak, sid, V_i, η) to the adversary $\mathcal{S}$.
Upon receiving (VoterID, sid, V_i, w_i) from $\mathcal{S}$,
if $\mathsf{V}_i \in \mathbb{V}_{\mathsf{corrupt}}$, then set Tab$[i] := w_i$;
otherwise, if $\mathsf{V}_i \in \mathbb{V}_{\mathsf{honest}}$ and $\eta = 0$, then set Tab$[i] := \perp$;
else, generate random temporal ID $w'_i \leftarrow \{0,1\}^\lambda$, and set Tab$[i] := w'_i$.

Voting/Delegation:

(1) Upon receiving input (Delegate, sid, V_j) from the voter $\mathsf{V}_i \in \mathbb{V}$, if $|T_1| < k$, ignore the input. Otherwise, record $B_i := (\text{Tab}[i], \mathtt{delegate}, \text{Tab}[j])$; send a message (ExecuteNotify, sid, V_i) to the adversary $\mathcal{S}$. If $|\mathbb{T}_{\mathsf{corrupt}}| = k$, then additionally send a message (Leak, sid, V_i, Delegate, V_j) to the adversary $\mathcal{S}$.

(2) Upon receiving input (Vote, sid, v_i) from the the voter $\mathsf{V}_i \in \mathbb{V}$, where $v_i \in \mathcal{V}$ and $\mathcal{V}$ contains all possible votes, if $|T_1| < k$, ignore the input. Otherwise, record $B_i := (\text{Tab}[i], \mathtt{vote}, v_i)$; send a message (ExecuteNotify, sid, V_i) to the adversary $\mathcal{S}$. If $|\mathbb{T}_{\mathsf{corrupt}}| = k$, then additionally send a message (Leak, sid, V_i, Vote, v_i) to the adversary $\mathcal{S}$.

Tally:

(1) Upon receiving input (Mix, sid) from the the trustee $\mathsf{T}_j \in \mathbb{T}$, set $T_2 := T_2 \cup \{\mathsf{T}_j\}$. Send a notification message (MixNotify, sid, T_j) to the adversary $\mathcal{S}$. If $|T_2| = k$, do
   - For $i \in [n]$, if B_i is not defined, set $B_i := (\perp, \mathtt{vote}, \perp)$.
   - Sort $(B_1, \ldots, B_n)$ lexicographically to form a new ballot vector $(\tilde{B}_1, \ldots, \tilde{B}_n)$.

(2) Upon receiving input (Tally, sid) from the trustee $\mathsf{T}_j \in T_2$.
Otherwise, set $T_3 := T_3 \cup \{\mathsf{T}_j\}$.
Send a notification message (TallyNotify, sid, T_j) to $\mathcal{S}$.

(3) If $|T_3 \cap \mathbb{T}_{\mathsf{honest}}| + |\mathbb{T}_{\mathsf{corrupt}}| = k$,
send (Reveal, sid, $(\tilde{B}_1, \ldots, \tilde{B}_n)$) to $\mathcal{S}$.

(4) If $|T_3| = k$,
compute *result* $\leftarrow$ TallyProcess($(\tilde{B}_1, \ldots, \tilde{B}_n)$, Tab).

(5) Upon receiving input (ReadResult, sid) from a voter $\mathsf{V}_i \in \mathbb{V}$, if *result* $= \emptyset$, ignore the input.
Else, return (ResultReturn, sid, *result*) to V_i.

Figure 1: The voting functionality.

actually open the content. However, surprisingly, this subtle issue was rarely modeled in the literature. For instance, the only previously known e-voting functionality [7] fails to address it. During the tally phase, the tally will be released if all the trustees agree to proceed. The formal description of the ideal functionality is depicted in Fig. 1 and Fig. 2.

TallyProcess

Input: a set of ballots $\mathcal{B} := (B_1, \ldots, B_n)$ and a set of voter labels $\mathcal{L} := (\mathsf{V}_1, \cdots, \mathsf{V}_n)$

Output: the tally result *result*

Label Check:

- Check the uniqueness of each label in $\mathcal{L}$. If there exist $i_1, \ldots, i_t \in [n]$, $i_1 \neq i_2 \neq \cdots \neq i_t$ s.t. $V_{i_1} = V_{i_2} = \cdots = V_{i_t}$, then replace all the labels $V_{i_1}, \ldots, V_{i_t}$ in $\mathcal{B}$ with $\perp$.

Delegation statement interpretation:

- For each ballot $B_i \in \mathcal{B}$, parse B_i in form of either $(\mathsf{V}_i, \texttt{delegate}, \mathsf{V}_j)$ or $(\mathsf{V}_i, \texttt{vote}, v_i)$ and do the following:
  - If B_i is in form of $(\mathsf{V}_i, \texttt{delegate}, \mathsf{V}_j)$ and $\mathsf{V}_j \neq \perp$, try to locate a ballot B_j in form of $(\mathsf{V}_j, \cdot, \cdot)$. If founded, replace $B_i := B_j$; otherwise, if B_i is in form of $(\mathsf{V}_i, \texttt{delegate}, \perp)$ set $B_i := (\mathsf{V}_i, \texttt{vote}, \perp)$.
  - Repeat the above step, until B_i is in form of $(\cdot, \texttt{vote}, \cdot)$. If there is a delegation loop, define $B_i := (\perp, \texttt{vote}, \perp)$.
- Denote the processed ballot set as $\mathcal{B}' := (B'_1, \ldots, B'_n)$.

Tally computation:

- Compute and return *result* $\leftarrow \mathsf{TallyAlg}(B'_1, \ldots, B'_n)$, where $\mathsf{TallyAlg}(\cdot)$ is the original tally algorithm.

Figure 2: The extended tally processing algorithm.

4 OUR DESIGN

Our toy example shows that it is possible to interpret the delegation statement by extending the tally algorithm. However, it is not immediately obvious about how to apply the same technique in conjunction with privacy. Before the voting/delegation phase, each voter can pick a temporal ID. However, the main challenge here is to distribute the temporal ID to the ones who need. The same as all existing end-to-end verifiable e-voting schemes, our design requires a publicly accessible consistent bulletin board, modeled as global functionality $\bar{\mathcal{G}}_{\mathrm{BB}}$. We let the voters post the re-randomizable RCCA encryption of their temporal ID on the $\bar{\mathcal{G}}_{\mathrm{BB}}$. If voter V_i wants to delegate his ballot to voter V_j, he can include a re-randomized ciphertext of V_j's temporal ID. More specifically, V_i sets his ballot as $B_i := (w_i, \texttt{delegate}, W_j)$, if he wants to delegate to V_j; or he sets $B_i := (w_i, \texttt{vote}, v_i)$, if he wants to vote for v_i; here w_i is V_i's temporal ID and W_j is the re-randomized ciphertexts of V_j's temporal ID. All the ballots will first be shuffled via a mix-net, and then all the trustees will jointly open those re-randomized ciphertexts inside the ballots. Subsequently, we can handle the delegation statement and compute the tally in the same way as the toy example. In the full version, we also show how to implement the liquid democracy voting scheme with re-randomizable threshold public key encryption (such as threshold ElGamal encryption) together with some necessary non-interactive proofs.

5 DISCUSSION AND FURTHER REMARKS.

In this work, we initiate the study of statement voting and liquid democracy. We remark that our statement voting concept can be significantly extended to support much richer ballot statements. It opens a door for constructing a new class of e-voting schemes. We also remark that this area of research is far from being completed, and our design and modeling ideas can be further improved. For example, if there is a delegation loop in which a set of voters delegate their votes to each other while no one votes, then what should be the "right" policy? Should the ballots be reset as blank ballots? This might not be ideal in reality. One possible approach is to extend the delegation statement to include a default vote. When a delegation loop exists, the involved ballots could be counted as their default votes. We finally remark that, voting policies can be heavily influenced by local legal and societal conditions. How to define "right" voting policy itself is a very interesting question. We believe our techniques here have the potential to help people to identify suitable voting policies which can further eliminate the barriers to democracy.

We remark that most non-trivial functionalities (including the e-voting functionality) cannot be UC-realized in the plain model [3–5]. Typically, we need trusted setup assumptions for UC secure e-voting systems, and the Common Reference String (CRS) model and the (Random Oracle) RO model are two common choices. In practice, if an e-voting system uses CRS, then we need to trust the party (parties) who generate(s) the CRS, which, in our opinion, is a stronger assumption than believing no adversary can break a secure hash function, e.g., SHA3. Therefore, we will realize our liquid democracy voting system in the RO model.

Alternative approach is as follows: we first use multiple party computation to generate a CRS; then we construct liquid democracy voting system by using the CRS. This approach has previously been used for constructing anonymous cryptocurrency Zerocash [1]; please see Ben-Sasson et al's recent effort [2]. We remark that, this approach might be problematic for cryptocurrency systems: typically a cryptocurrency system will last for many years and it is very difficult to ensure there is no attack on the CRS during this long time period. Interestingly, this limitation does not apply to liquid democracy voting systems. If there is an issue with the current CRS, we can use MPC to generate a new CRS.

REFERENCES

[1] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 459–474.

[2] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. 2015. Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In *2015 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 287–304.

[3] Ran Canetti. 2001. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd FOCS*. IEEE Computer Society Press, 136–145.

[4] Ran Canetti and Marc Fischlin. 2001. Universally Composable Commitments. In *CRYPTO 2001 (LNCS)*, Vol. 2139. Springer, Heidelberg, 19–40.

[5] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. 2003. On the Limitations of Universally Composable Two-Party Computation without Set-up Assumptions. In *EUROCRYPT 2003 (LNCS)*, Vol. 2656. 68–86.

[6] Bryan Ford. 2002. Delegative Democracy. Manuscript. http://www.brynosaurus.com/deleg/deleg.pdf.

[7] Jens Groth. 2004. Evaluating Security of Voting Schemes in the Universal Composability Framework. In *ACNS 04 (LNCS)*, 46–60.