Opportunistic Vote Delegation for e-Voting based on Liquid Democracy

[Work in Progress]

Jó Ágila Bitsch Link, Angel Tchorbadjiiski, Klaus Wehrle*
COMSYS, RWTH Aachen University
Aachen, Germany
{jo.bitsch|angel.tchorbadjiiski|klaus.wehrle}@rwth-aachen.de

ABSTRACT

We present an approach for implementing the liquid democracy concept in a secure, anonymous and publicly verifiable manner via opportunistic networks. Liquid democracy is gaining traction in open government and civil participation, as it allows vote delegation and in a hybrid form of direct and representative democracy.

Combining RSA public-key cryptography with hash-based data structures, this system allows the acquisition of anonymous voting tokens, their delegation while still allowing outvoting, and secret, anonymous voting. Voters can verify their own vote, while the public can check the consistency of the overall voting results. The solution thereby enables the wider use of liquid democracy to further enhance civil participation in the government process, while having a high resilience against vote manipulation, i.e. manipulation will be discovered with a probability of higher than 99% with less than 1% of the voters verifying their vote.

1. INTRODUCTION

The concept of liquid democracy represents a mixture of both direct and indirect democracy for a decision-making process and allows every participant to how involved in this process she wants to be. For every election taking place, it is possible to either take part directly or delegate the own voting rights to a representative or an expert. This way, the voters are not limited to taking one decision for a legislative period as opposed to indirect (representative) democracy, but are able to actively and continuously take part in the decision-making process.

In this work, we present the—to the best of our knowledge—first e-voting scheme and prototype implementation of liquid democracy, that allows secure and anonymous voting, even against collaborating system administrators. Our scheme allows the revocable delegation of voting rights, using only local or opportunistic communication. Anonymity is provided through the use of blind signature, while after the election, the person actually using the vote, can verify that it was counted correctly. As in normal paper based elections, a possibility of cheating still exists. However, the detection rate, depending on the number of voters checking the correctness of their own votes, is high even for a small amount of checkers.

*This work has been cofunded by the DFG as part of the CRC 1053 MAKI.

ExtremeCom '14, August 11-15, 2014, Galapagos Islands, Ecuador.

1.1 Contributions

The main contributions of this paper are:

1. **e-Voting scheme for liquid democracy** - We present a scheme for secure and anonymous voting and vote delegation. The scheme is cryptographically secured even against system administrators.
2. **Opportunistic and local vote delegation** - Voters can delegate their voting credentials even in the complete absence of global connectivity using only local or opportunistic communication.
3. **Prototype implementation** - We provide a sample implementation that proves the practical feasibility of the presented voting scheme.

2. REQUIREMENTS

We aim at creating a voting architecture based on the liquid democracy concept. The focus is on the security of the system and the anonymity of the participating users, while allowing the secure delegation of votes, even in the absence of global network connectivity.

The solution should conform to typical voting regulations and laws in a democratic country. As an example, the German constitution [3, §28] requires, that every election for the parliament is *general*, *direct/immediate*, *free*, *equal* and *secret*. This leads to the following requirements, which have to be satisfied by the design:

1. **Voting over networks** - The voting procedure can be carried over an insecure network like the Internet. Data transfer is encrypted and integrity protected.
2. **Anonymity** - The information available in the system doesn't allow the identification of individual voters or the correlation between them and their votes.
3. **Voting results integrity** - Manipulation of voting data is not possible or highly improbable. The voting results are public and verifiable by everyone.
4. **Only authorized participants can vote** - The identity of the individual can be checked at the moment of issuing voting credentials and the right to vote can be verified during voting. The server can generate valid credentials only on request of an authorized user. If it randomly generates legit credentials, the chance of being detected is high.
5. **Vote delegation** - The own voting rights and delegated votes can be further delegated to one or more entities, albeit only a limited number of times. Delegation should also be possible using local or opportunistic communication.
6. **Delegation revocation** - A delegator can revise a vote forwarding decision at any time before the election is over.

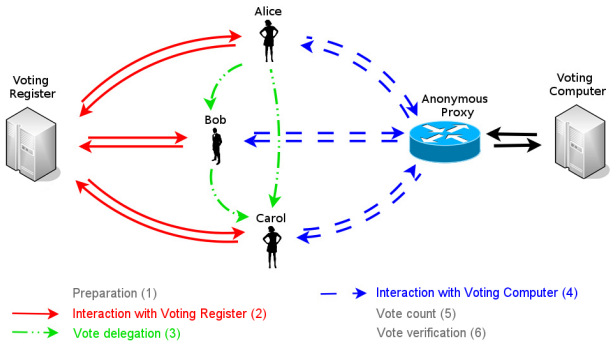

Figure 1: System design showing the relations between the entities.

The constitutional requirements *general*, *free* and *equal* are fulfilled through the anonymity-requirement (2) as this prevents the system to distinguish between different users. Fulfilling the requirements (1 and 2) makes it impossible for an observer with (insider) or without knowledge (outsider) of system information to know what a voter has voted for. Though bystanders, who are able to coerce/threaten the voter, can be present at the moment of voting. This is mitigated by the requirements (5 and 6), which would allow a repeated voting, but would suppose that the bystander is no longer on site. This problem also exists in the postal voting mechanism, which conforms to the German Constitution.

A major point of contention [2] is that the algorithms and cryptography needed for electronic voting systems in general might be too complex for the average voter to understand and therefore inadmissible for general elections in Germany.

3. RELATED WORK

While the concept of liquid democracy is gaining traction, there are no other systems that give cryptographic guarantees to participants. In the following, we will give a short overview of the strengths and weaknesses of used and proposed systems.

The main goal of **Adhocracy** [5], a python based web application, is the collaborative development of proposals and voting for consensus on them. To achieve this, different forms of delegation are available: *one-time*, *multiple* and *topic-delegation*. To allow verifiability of the voting, the complete voting history for every user is publicly available. "Adhocracy does not allow for any kind of secret voting. A full public record is available for all users at any time." From the documentation on Adhocracy [6, Secret Ballots]. While this design choice allows Adhocracy to be used to publicly and collaboratively develop ideas, final decisions necessitating secret ballots are inherently precluded. Furthermore, the reliance on a single web platform makes operation in low connectivity environments impossible. A related approach is **LiquidFeedback** [7]. The key difference is an additional hierarchical level to sort proposals. Every initiative has to pass through a set of statuses in a given time span and reach a predefined quorum level.

Votorola [1] provides a web interface similar to MediaWiki and uses the concept of communicative delegation, where a small group of participants with similar interests forms a draft suiting their common interests and delegate their voting rights to a representative. This happens as many times as needed to reach a reasonably small amount of drafts, so that a voting procedure can take place. Built on tree-based structure, Votorola supports vote delegation. Again, voting procedures are fully transparent, making it impossible to have anonymity or secrecy in the system.

The recent scheme by **Zwattendorfer et al., 2013** [9] allows secret votes, but also the secure anonymous delegation to a proxy. Their scheme relies on several servers: an (1) Election Server, which keeps the general information on an election and a list of responsible servers, a (2) Ballot Signer, which authenticates a voter and provides her with a signed ballot, and a (3) Voting Server, which collects the final votes. However, the voter needs to be online for the election, even if she delegated her vote, as the voting right is not transferred but the vote of a proxy is copied. If the Ballot Signer collaborates with the Election Server and keeps a copy of the vote (which is encrypted with the public key of the Election Server), the anonymity of the voter can be broken.

4. DESIGN

Our system uses a Voting Register (VR), a Voting Computer (VC) and an Anonymous Proxy (AP), cf. Figure 1. The VR contains all the information needed to verify the identity of the voters, supplies them with voting credentials and records who did get credentials and who did not. The VC is responsible for verifying the validity of the voting credentials, recording the ballot and issuing a receipt for it. The AP is needed so that the VC is not able to tell voters apart (relying for instance on IP address information) and not able to discriminate them.

The necessary steps for one voting round are: (1) **Preparation** - The VC and the VR generate new private and public key pairs, publish their public keys and reinitialize their databases. The VR receives a fresh eligible voter list. (2) **Interaction with Voting Register** - A voter authenticates with the VR and presents a blinded authentication token based on a fresh hash chain. The VR signs the token and returns it to the voter, who unblinds it for later use with the VC. (3) **Vote delegation** - A voter can delegate her vote by revealing an element of her hash chain to another voter. The distance to the anchor of the hash chain corresponds to the priority of the delegation and allows later revocation. (4) **Interaction with Voting Computer** - The voter submits her and any delegated votes to the VC. The VC signs this transaction with its key, so that the voter can later verify that her ballot has been considered in the final count. (5) **Vote count** - After the closing of the vote, the VC tallies up all ballots and publishes the results, together with lists of additional information, which allows later verification. (6) **Vote verification** - A voter can verify her ballot using the published lists from the VR, VC and the receipt the VC supplied during step 4.

4.1 Preparation

Both VR and VC generate fresh RSA key pairs and reinitialize their internal databases. This explicitly invalidates all voting credentials issued in former rounds. Then, the VR acquires a fresh copy of the register containing all citizens and their public keys from the responsible authorities. This list is signed with the fresh private key of the VR and both the signed list version and its public key are made public.

Publishing the signed register list allows citizens to verify whether they are on the list and if the appropriate public key is saved for them. As the VR can generate credentials only for users on this signed list, the upper limit of voters and their keys are publicly known. If the VR manipulates the list and modifies a public key for a person, this person will, with high probability, detect the changed key and can prove that the VR tried to cheat. The possibility to acquire voting credentials is time limited. After this period, the VR has to publish the list of all citizens, who have acquired voting credentials. This way there is no possibility to add additional credentials later and the number of voters is delimited again.

Each voter generates a hash chain [4] through the following procedure: (1) Calculate a random starting point $h_S = h(\text{rand}())$ with the help of a cryptographic hash function. (2) Use h_S to calculate a hash chain HC with an *anchor* element h_A and depth D. Save the elements h_S and h_A as they define the hash chain HC.

VR	Voting Register
VC	Voting Computer
AP	Anonymous Proxy
AMB	Anonymous Message Board
$h(\cdot)$	Cryptographic hash function
rand()	A random number generator
HC	Hash chain
h_A	Anchor of a hash chain
h_S	Generator (Source) of a hash chain
h_I	Intermediate element of a hash chain, which can be used for voting
D	Length (Depth) of a hash chain
$\Anchor$	The signed anchor of a hash chain $(h_A, \text{sig}_{\text{VR}}(h_A))$
r_B	A random number used for blinding
$[\cdot]_{r_B}$	The blinded value, with blinding factor r_B
$\text{sig}_{\text{VR}}([h_A]_{r_B})$	The blind signature of VR on h_A
h_{Vote}	The intermediate hash element h_I, actually used for voting
nonce	A nonce used during the vote submission
vote	The vote a user makes

Table 1: List of symbols and abbreviations used in this paper

As HC is generated with a cryptographic hash function, only the user with the starting element is able to recreate it. Having an element h_I between h_S and h_A allows the generation of the part (h_I, h_A), but does not divulge information about preceding elements. This characteristic of hash chains is used for the delegation. The starting first element is kept secret, other elements can be delegated. The distance from the anchor defines the priority of the delegated votes.

4.2 Interaction with Voting Register

After the initialization procedures, communication with the VR is possible and every user on the register list can acquire voting credentials. The voter initiates an encrypted and integrity protected connection to the VR and authenticates using her private key. This allows the VR to check if the user is allowed to acquire voting credentials and did not do that already.

To make it possible for the VC to verify that the credentials supplied are valid, a signature from VR on the anchor element of the voter is needed. As the voting procedure has to be anonymous, no connection between the voter's identity and the credentials used should exist. To ensure this, Alice's anchor h_A is blindly signed [8, Chapter 5.3] through the following procedure:

(1) Alice generates a random value r_B for blinding and uses it in combination with the public key of VR to scramble h_A producing $[h_A]_{r_B}$. (2) Alice signs the triple $([h_A]_{r_B}, \texttt{TS}, \texttt{ID})$ with her private key resulting in $\text{sig}_{\text{Alice}}([h_A]_{r_B}, \texttt{TS}, \texttt{ID}_{\text{Alice}})$, where TS is the current time stamp and $\texttt{ID}_{\text{Alice}}$ the identity of Alice. (3) Alice sends $([h_A]_{r_B}, \texttt{TS}, \texttt{ID}_{\text{Alice}}, \text{sig}_{\text{Alice}}([h_A]_{r_B}, \texttt{TS}, \texttt{ID}_{\text{Alice}}))$ to VR. (4) The VR checks the signature. If it is successful, it saves the value into its database, signs $[h_A]_{r_B}$ and returns the resulting $\text{sig}_{\text{VR}}([h_A]_{r_B})$ to Alice. (5) The VR makes $([h_A]_{r_B}, \texttt{TS}, \texttt{ID}_{\text{Alice}}, \text{sig}_{\text{Alice}}(\cdot))$ and $\text{sig}_{\text{VR}}([h_A]_{r_B})$ available in its list for later verification as a receipt that Alice got her credentials. (6) Alice in turn can unblind the signature using r_B. The resulting $\Anchor = (h_A, sig_{\text{VR}}(h_A))$ and its generator h_S or any intermediate element h_I presents the anonymous voting credentials of Alice for the VC.

The signed receipt $([h_A]_{r_B}, \texttt{TS}, \texttt{ID}_{\text{Alice}}, \text{sig}_{\text{Alice}}(\cdot))$ is used to preclude VR from issuing randomly generated voting credentials. As the triple contains a fresh time stamp TS and the actually used blinded value and only Alice is able to generate a valid signature on it, she has actively participated in the protocol, so no cheating by VR is possible. After signing $[h_A]_{r_B}$, the VR transmits the result to Alice and makes the signed triple public. As Alice's public key is available in the voters list, everyone can check the triple.

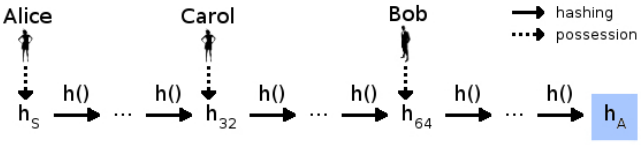

Figure 2: Alice first delegates the credentials $(h_{64}, h_A, sig_{VR}(h_A))$ to Bob, then she reconsiders her decision and supplies Carol with the higher priority credentials $(h_{32}, h_A, sig_{VR}(h_A))$.

4.3 Vote Delegation

To demonstrate the delegation procedure consider the users Alice, Bob and Carol. Alice, who is in possession of the hash chain $HC_{\text{Alice}} = (h_S, \Anchor)$, decides to delegate her voting rights to both Bob and Carol. The closer a forwarded h_I is to h_S, the higher is its priority. The result of this process is presented in Figure 2.

Users can forward credentials using a *secure* and possibly *anonymous* channel, i.e., local communication via NFC or Bluetooth, encrypted e-mail, opportunistic communication, encrypted posting on an Anonymous Message Board, randomly generated URL on a private web server or any other side channel.

4.4 Interaction with Voting Computer

After acquiring credentials by the VR or through proxying, Alice can connect to the VC and take part in the voting procedure. Alice may use an anonymous proxy (AP) to connect to the VC. This way her identity cannot be determined through her IP address.

Next an encrypted and integrity protected connection is initiated between Alice and the VC. This way the AP is not able to read any plaintext traffic although it is in a Man in the Middle position. To start the authorization procedure Alice sends $\Anchor$ to VC. Through the use of blind signatures by the VR, Alice stays anonymous. Through a DHKE [8, Chapter 22], Alice and VC negotiate a shared nonce. This nonce is only known to the actual voter so if published together with the voting decision, it allows verification and does not break the secrecy of the voting.

Alice concatenates the shared nonce with her voting decision vote to $(\texttt{nonce}, \texttt{vote})$. With the help of VC's public key she blinds $(\texttt{nonce}, \texttt{vote})$ using the starting hash $h_{\text{Vote}} = h_I$ of her hash chain as a secret and transfers the resulting blinded string $[(\texttt{nonce}, \texttt{vote})]_{h_{\text{Vote}}}$ to VC. Through this procedure the risk of VC filtering based on the choice of a voter is mitigated, as VC commits to the choice without seeing it and moreover still has no access to the actual voting credentials (only the anchor is known).

VC saves the value $[(\texttt{nonce}, \texttt{vote})]_{h_{\text{Vote}}}$ temporarily (until the end of connection), signs $[(\texttt{nonce}, \texttt{vote})]_{h_{\text{Vote}}}$ with its private key (commitment) and returns the result to Alice. She reverses the blinding and acquires a signature of the vote she would like to submit. As the VC can't read the submitted vote, she has to send it the value h_{Vote} for the vote to become valid.

Now that the VC is in possession of the h_{Vote}, it first checks if the hash chain was not already used for voting. If this is not the case VC verifies that h_{Vote} is really the secret used for blinding and also that it allows the generation of the signed anchor presented for authorization in a maximum of n_{max} steps. If all this is the case, the voting values are checked for correctness - the value nonce has to be equal to the one generated on the server side and the vote has to be contained in the list of voting options. When this last check

VR	Eligible Voters	$\texttt{ID}_i$
	Collected Tokens	$\texttt{ID}_i$,$([h_A]_{r_B}, \texttt{TS}, \texttt{ID}_i, \text{sig}_i(\cdot))$
VC	Used Tokens	h_{Vote_i},⫫_i
	Votes	$\texttt{nonce}_i$,$\texttt{vote}_i$
Voter	VC Receipt	sig_{VC} $((\texttt{nonce}, \texttt{vote}))$
	Vote Token	h_{Vote}, ⫫

Table 2: List of tables used for vote verification. The VR and the VR list are sealed with their respective private keys.

is satisfied, a new receipt depending on the last issued receipt is generated with the help of another cryptographic hash function, all information (meaning: h_{Vote}, ⫫, nonce, vote, receipt) is saved in the database and the receipt it returned to the user. The (h_{Vote},⫫) values are added in a black list. This way voting with the same credentials is no more possible.

4.5 Vote Count

After the period assigned to the voting has elapsed, the results have to be sealed. This way no further modification is possible. For this purpose two lists are generated by VC, signed with its private key and made publicly available. The first contains all the tuples (h_{Vote},⫫). Its purpose is to allow everyone to check, if only unique anchors are used and verify the corresponding signatures. Furthermore, the voting hash to anchor relation (the hash chain) can be tested. The second list consists of the (nonce,vote) pairs. This allows voters to check if their submitted values match the recorded ones, although no connection between vote and (h_{Vote},⫫) tuple is publicly present. The finalization process ends when the saved data is signed by a trusted third party and is copied to a secure location.

4.6 Vote Verification

To detect cheating by the VR, we have the following information, cf. Table 2. The voter's identity and the corresponding RSA public key are available from the public voter list. As the VR publishes a list of who requested voting credentials and keeps the signed (blinded) request, a ballot verifier can verify that only real voters acquired voting credentials.

To detect cheating by the VC, we can check the following properties: The h_{Vote} of a vote must match the signed anchor ⫫ after a maximum number of hashes. The signature on ⫫ must match the public key of VR. IOn revocation, the lesser priority voter recognizes this, as the published h_{Vote} hashes to her own h_{Vote}.

A voter whose delegation was not revoked can verify her own vote in the (nonce,vote) list. The other verifiers can calculate the overall voting results from this data and assure no modification or cheating has taken place.

5. EVALUATION

As algorithms and systems for liquid democracy are still a relatively young research area, best practices in evaluation are still emerging. In the following, we give a short qualitative comparison to related work and current weaknesses. A prototype implementation is available online at https://bitbucket.org/scapy/liquid_democracy_concept.

Table 3 juxtaposes our system with the related work. Our system lacks features like the modular structure of Votorola, the discussion features and time/topic delegation, but allows publicly verifiable voting results without compromising the anonymity and secrecy of the ballots. Furthermore, it makes it nearly impossible to modify information in the system without this being noticed. Zwattendorfer et al.'s system depends on the different involved servers not cooperating to keep the ballot secret and the voter being online for the vote, even when she delegated the vote. Our system avoids this using blind signatures and transferring revocable voting credentials to the delegate instead of copying the vote of the delegate.

	A	L	V	Z	S
multiple delegation	✓	✓	–	✓	✓
time/topic delegation	–	✓	–	–	–
discussion possibilities	✓	✓	✓	–	–
public results	✓	✓	✓	✓	✓
voting receipt	–	–	–	✓	✓
modification detection	-	-	-	+	++
anonymous/secret ballot	–	–	–	o	✓
cryptographically secured	–	–	–	o	✓
modular structure	✓	–	✓	–	–
distributed setup	–	–	✓	✓	✓

Table 3: Comparison Adhocracy (*A*), LiquidFeedback (*L*), Votorola (*V*), Zwattendorfer et al. (*Z*) and our System (*S*)

To illustrate the resilience against manipulation, we use a ballot with 44 million voters, the number of votes submitted in the German general elections in 2009. Even with 20.000 manipulated votes (≈*0.5 %* of all voters) less than 10.000 verifiers (≈*0.25 %* of all voters) would be needed to reliably detect cheating.

A major possible weakness of the presented approach is the possibility of vote buying or coercion. As the voting results are public and all participants can check their own votes, a coercing party even has proof, if a victim voted accordingly. Another possible weakness is the privacy of non-voters: As the VR publishes the information, who picked up their voting credentials, voters might on the one hand be coerced into not voting at all or on the other hand publicly shamed for not voting.

6. CONCLUSIONS

We presented an e-voting scheme for liquid democracy, which allows the cryptographically secure delegation of votes, even when the delegator only has local or opportunistic communication available. A voter can submit her vote completely anonymous and still verify her vote. Therefore, this work lays the foundation to bring electronic version to disconnected regions.

7. REFERENCES

[1] ALLAN, M. Votorola. Online Resource. http://zelea.com/project/votorola/home.html, accessed 2014-05-18.

[2] FEDERAL CONSTITUTIONAL COURT OF GERMANY. Verwendung von Wahlcomputern bei der Bundestagswahl 2005 verfassungswidrig. Online Resource, 03 2009. http://www.bundesverfassungsgericht.de/pressemitteilungen/bvg09-019.html, accessed 2014-05-18.

[3] GERMAN BUNDESTAG. Constitution of the Federal Republic of Germany. Online Resource. http://www.iuscomp.org/gla/statutes/GG.htm, accessed 2014-05-18.

[4] LAMPORT, L. Password authentication with insecure communication. *Communications of the ACM* (1981), 770–772.

[5] LIQUID DEMOCRACY E.V. What is liquid democracy? Online Resource. https://adhocracy.de/static/about_adhocracy/what_is.html, accessed 2014-05-18.

[6] LIQUID DEMOCRACY E.V. Adhocracy wiki - secret ballots. Online Resource, 2014. http://trac.adhocracy.de/wiki/SecretBallots, accessed 2014-05-18.

[7] PUBLIC SOFTWARE GROUP E.V. Liquidfeedback. Online Resource, 2012. http://www.public-software-group.org/liquid_feedback, accessed 2014-05-18.

[8] SCHNEIER, B. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. 1995.

[9] ZWATTENDORFER, B., HILLEBOLD, C., AND TEUFL, P. Secure and privacy-preserving proxy voting system. ICEBE '13, pp. 472–477.