Secure and Privacy-Preserving Proxy Voting System

Bernd Zwattendorfer, Christoph Hillebold, and Peter Teufl
Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Austria
{bernd.zwattendorfer, peter.teufl}@iaik.tugraz.at, christoph.hillebold@student.tugraz.at

Abstract— **Voting is a frequent and popular decision making process in many diverse areas, targeting the fields of e-Government, e-Participation, e-Business, etc. In e-Business, voting processes may be carried out e.g. in order management, inventory management, or production management. In this field, voting processes are typically based on direct voting. While direct voting enables each eligible voter to express her opinion about a given subject, representative voting shifts this power to elected representatives. Declarative or proxy voting (based on liquid democracy) is a voting process situated in between these two approaches and allows a voter to delegate her voting power to a so called proxy, who actually casts the votes for all the represented voters. The most interesting aspect of this approach is that voters have the opportunity to skip the direct involvement when they trust the proxy to act within their best interest. Liquid democracy and proxy voting has been implemented in various software tools that facilitate the voting process. However, the current systems lack security features typically required by electronic voting systems. Therefore, we present a system that integrates cryptographic functionality and relies on qualified signatures created by the Austrian citizen card to solve the current security issues. This system can support e-Business processes and applications in decision making, enabling the delegation of votes.**

Keywords- liquid democracy, proxy voting, Austrian citizen card, security, strong authentication, e-voting

I. Introduction

Voting is a frequent and popular decision making process in many diverse areas, targeting the fields of e-Government, e-Participation, e-Business, etc. In e-Business, voting processes may be carried out e.g. in order management, inventory management, or production management. Liquid democracy [1] [2] is a method that can be used for decision-making, also in e-Business.. The most interesting property – when compared to most conventional electronic voting systems – is the capability to allow users (voters) to delegate their voting power to others. In liquid democracy, decision-making also includes discussions, finding election issues, and holding elections [2] [3]. In our work, the main emphasis is placed on proxy voting, which deals with the aspects of direct voting or vote delegation.

In general, there are two kinds of users in a proxy voting system [1]:

Voter: A voter is a user of the system, who is allowed to vote for elections. The voter could either vote directly on an election or delegate her voting power to another user called proxy. A voter must typically vote secretly.

Proxy: A proxy is a voter that wants to get delegations from other voters. Delegations are kept secretly and are not public. Like a user, also a proxy could either vote directly or delegate her voting power to another proxy. A proxy can be compared with a politician whose opinion must be public. Therefore a proxy cannot vote secretly and has to publish her vote.

Proxy voting allows voters either to vote directly or to delegate their voting power to a proxy. Delegations could be solved in two ways:

1. Either the voter copies the published vote of the chosen proxy (client-based) or
2. the voter delegates her voting power permanently to the proxy (server-based).

In addition, users, who delegated their voting power to a proxy, might change their opinion and thereby deviate from the proxy's behavior. Even more problematic, a proxy could change her mind just before the election process and thus cast a different vote than her users expected. Hence, voters should be able to revoke their delegation and vote by themselves.

Several proxy voting systems supporting the described functionality already exist and are described in Section II. However, all of these systems usually rely on web-based solutions deployed on a single server. The drawback of a single server solution is that – if this server fails – the whole system will be compromised. Additionally, the use of simple web browsers is critical as they usually do not support required cryptographic functions to securely use the proxy voting system out of the box. Furthermore, all of those systems do not support unique identification and authentication. This requirement is particularly essential to avoid casting multiple votes by a single person. A recent demonstration by a German journalist [4] shows that existing systems do not always fulfill these requirements. Thereby, she was able to create two accounts within the Liquid Feedback system (see Section II) and was able to vote twice.

To bypass these issues, we propose a new architecture for a proxy voting system, which relies on multiple servers. In addition, instead of a web browser we rely on a Desktop-based application, which integrates the required cryptographic functionality to make our proxy voting system secure. Finally, we employ qualified signatures by using use the Austrian citizen card to uniquely identify citizens and thus to avoid multiple voting possibilities. Although our system identifies citizens uniquely, our distributed architecture relying on multiple servers, and the implemented cryptographic processes allow us to preserve citizens' privacy and support anonymous voting.

II. Related Work

In this section we briefly describe related work dealing with liquid democracy and proxy voting. Other communication tools to be used in liquid democracy can be found in [5]. All subsequent projects are single instance solutions, web-based, and require a web browser as user client. For authentication simple username/password schemes are applied. While they basically fulfill the functional requirements of proxy voting systems, they are not able to meet all security requirements identified in Section III.

A. *Votorola*

Votorola[1] is a liquid democracy project published by "zelea.com". Votorola is based on a Wiki platform, where any user can create and modify drafts. Users can discuss and vote for changes of those drafts, and – after a certain period – finally only some of the original drafts survive. At the end, the draft or user with the most votes or supporting delegates wins. If a user possesses more than one vote, separation of the voting power is not supported. She rather has to delegate all her votes to one single proxy at once. The architecture of the Votorola project is designed in a modular way. I.e., single modules (e.g. authentication module) can easily be replaced [6]. Currently, authentication is implemented using OpenID[2] or via e-mail, where the e-mail-address is published on the website. One main feature of Votorola constitutes the replication of votes. Here, votes can be replicated between several Votorola systems with different modules for data protection and backup purposes. However, it is also possible to replicate votes to an instance of another liquid democracy system such as Adhocracy [7], which will be described next.

B. *Adhocracy*

Adhocracy[3] is an open source liquid democracy project developed by the association "Liquid Democracy e.V."[4]. In general, Adhocracy is a free participation platform enabling organizations, its members, and any interested citizen the possibility for an open and transparent democratic communication. Additionally, Adhocracy offers citizens an information platform to several activities, discussions, or decisions of organizations. To achieve this vision of easy civic participation, the main pillars of Adhocracy are transparency, autonomy, and modularity [8]. Referring to the objective of transparency, all decisions or votes are transparent to arbitrary users all the time. In addition, all discussions or articles are publicly available. Referring to autonomy, all groups or discussion forums are managed autonomically and do not require a group manager. Proposals or comments to individual topics are rated by the members to evaluate their relevance. Finally, Adhocracy provides modularity to organizations. Groups and forms of decision making can be easily customized to individual requirements.

[1] http://zelea.com/project/votorola/home.html

[2] http://openid.net

[3] https://adhocracy.de

[4] https://liqd.net

C. *Liquid Feedback*

Liquid Feedback[5] constitutes also an open source platform enabling decision making based on liquid democracy. Liquid Feedback is developed by the "Public Software Group". In general, Liquid Feedback respects the following concepts [9]: liquid democracy (votes can be delegated by topic), proposition development process (return structured feedback for an initiative), preferential voting (users can state preferences instead of simple yes/no votes), and interactive democracy (use of interactive electronic media). Liquid Feedback can be used by several entities and for several use cases. For instance, political parties, associations, NGOs, governments, or even corporate bodies rely on the functionality of Liquid Feedback.

III. Requirements

In this section we summarize the requirements which have to be met by a secure and privacy-preserving proxy voting system. These requirements are aligned to requirements of conventional electronic voting systems [10] [11]. The main difference of proxy voting systems compared to conventional electronic voting systems is the support of vote delegation.

A. *Functional Requirements*

Voting: The most important feature of every electronic voting system is the voting process itself. Voters can vote for an election and cast their ballot. The voter must not be able to vote twice for the same election.

Vote Delegation: Proxy voting requires the ability to delegate the voter's voting power to a proxy. A user, who delegated her voting power to a proxy, is not allowed to cast her own vote in the respective election. The voting power of the chosen proxy is (virtually) increased. We have identified two basic principles how delegation of votes can be achieved. A proxy voting system should at least support one of these principles.

1. Server-based delegation: In this model, vote delegation is carried out via a server. Basically, the voter selects a proxy for delegation and encodes the delegation information similar to a conventional vote. If the proxy casts her vote, the users' delegated votes automatically count for the same answer as the proxy's vote. A proxy could also delegate her voting power to another proxy transitively. The advantage of this approach is that the user's voting power can be delegated even if the user is not online. However, the disadvantage of this approach is that the proxy could change her mind without notice of the user. The user would notice such a change only if she has an online connection to the server. In addition, it is possible to find out how many delegations a proxy has, because this information has to be stored on the server. This could support corruption and blackmail because the voting power of the proxy could be made public.

2. Client-based delegation: In this approach, vote delegation is carried out on the user's client. The voter selects a proxy

[5] http://www.public-software-group.org/liquid_feedback

to delegate her votes and stores the information locally. After the proxy has published her vote, the voter can download and copy it. This requires the voter to be online at least once to set the vote during an election period. The advantage of this approach is that it is not possible to find out who is voting for which proxy as every delegation is made locally. Additionally, the voter is able to intervene and change her decision any time by revoking the delegation locally. However, the downside of this approach is that delegation cannot be placed automatically such as in the server-based approach.

Rejection and Revocation of Votes: If a voter does not agree anymore with the opinion of her proxy, she should be able to revoke the delegation of her vote until the end of the election. After revocation, the user's old delegation or vote becomes invalid and she will be able to vote another time again. If the system relies on the server-based delegation approach, then the deadline for proxies being able to change their vote must be some time before the actual end of the election. Voters then have sufficient time to revoke their delegation and vote directly or delegate their vote to another proxy. In the client-based delegation approach, voters can revoke their delegation locally as long as the election time frame is open.

Determine the Election Winner: There are several models to count the votes and find out the winner of an election. Note that within liquid democracy the winner is a specific answer to an election and not a politician. In a liquid democracy proxy voting system, different methods for determining an election winner are available [12]. We just highlight three, which we consider most important.

1. *Plurality Voting System:* In this model, the answer receiving the most votes wins.
2. *Preferential Voting System:* In this model, the voter can state preferences for each answer. Still, the answer with the most weighed votes wins.
3. *Two-round Voting System:* In the first round, more than two possible answers are available, which can be voted for. After the first round, only the two answers having received the most votes remain. In the second round, only two answers can be voted for and again the answer with the most votes wins.

B. Security Requirements

Anonymity: Users must be able to vote anonymously all the time. In addition, user must not be linkable by any other means.

Secrecy: Nobody should be able to see the content of a ballot until the end of the election. Otherwise, voters could follow the proceeding of the election and use the information to manipulate the result either actively by placing the own vote in dependence of the current situation or passively by manipulating others through statistics.

Integrity: No single entity of the proxy voting system should be able to manipulate the system by generating votes without legitimation, modifying valid votes, deleting valid votes, invalidating votes, rejecting votes without legitimation, using circular transactional delegations or rejecting or voting after the deadline.

Authenticity: Only people that are allowed to vote should actually be able to vote and no user should be able to vote more than once per election. Therefore, users must be uniquely identified and securely authenticated by the voting system.

Verifiability: Everyone should be able to audit and to verify if the voting system works correctly. Thereby, users should be able to check if their own votes are still present in the voting system. Additionally, users (by using their client) must be able to count all valid votes to verify the official result. Therefore, all votes – not identifiable or linkable to a certain user – must be publicly readable after the election.

IV. System Architecture

In this section we propose a new architecture for a proxy voting system to meet the identified requirements. The architecture consists of at least three separated components: one *Election Server*, one or more *Ballot Signers*, and multiple *Voting Servers*. Users access the proxy voting system via a *User Client*. Fig. 1 illustrates an overview of our proposed proxy voting system architecture. Solid arrows symbolize network connections with a server considered to be trustworthy. The dashed arrow between the *User Client* and the *Voting Server* also symbolizes a network connection, but the *User Client* may not trust the *Voting Server*.

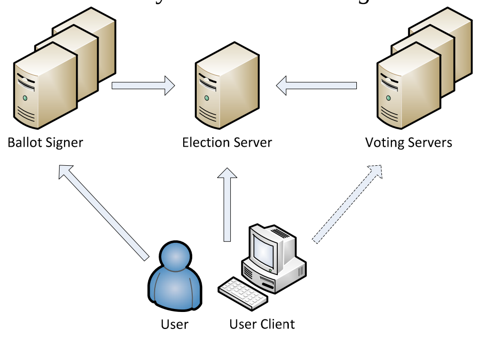

Fig. 1. Architecture of the proxy voting system

A. Components of the System Architecture

In the following, we describe our architecture and the individual components in more detail.

Election Server: The *Election Server* is mainly responsible for providing general information to the individual users. For instance, this includes information on what elections are active and when the elections will end. Additionally, the *Election Server* publishes the official results of ended elections. The *Election Server* also manages a table or database of all available *Ballot Signers* and *Voting Servers*.

Ballot Signer: The *Ballot Signer* authenticates the user and checks whether the user is allowed to take part at a certain election. Also, the *Ballot Signer* validates the user's vote without being able to inspect the user's decision. It is possible to set up and operate more than one *Ballot Signer*, e.g. one for each federal state of a country or one for each organization that is allowed to participate in the system. This may help in making regional or partial statistics and limits the power of on single *Ballot Signer*.

Voting Servers: The *Voting Servers* have to store encrypted votes, which have been issued in encrypted format by the *User Client*. The encrypted vote is actually not secret but cannot be linked to a specific user. Therefore, anybody could set up and publish her own *Voting Server*. The *Voting Servers* count all valid votes. If a *Voting Server* accepts a vote from a user, the vote is signed by the *Voting Server* and returned to the user. Hence, it is not possible for a *Voting Server* to delete a vote without detection, because multiple *Voting Servers* store the same vote and the user has a proof (the vote signed by the *Voting Server*) that the *Voting Server* has accepted the vote.

B. Supported Functionality

To illustrate the functionality of our proxy voting system, we describe relevant processes to be carried out in a system supporting delegation of votes. In particular, we describe the voting process itself, the revocation of votes, counting of votes, and the process of verifying the system.

1) Voting and vote delegation

Basically, the aim of the voting-process is to publish a valid vote. To do that, the user must be authenticated by the *Ballot Signer*, which will sign the vote. Then the *User Client* can distribute the signed vote to other *Voting Servers*.

Fig. 2 shows a detailed sequence diagram of the complete voting and delegation process involving all components. We will describe the individual process steps in the following taking into account the numeration of Fig. 2.

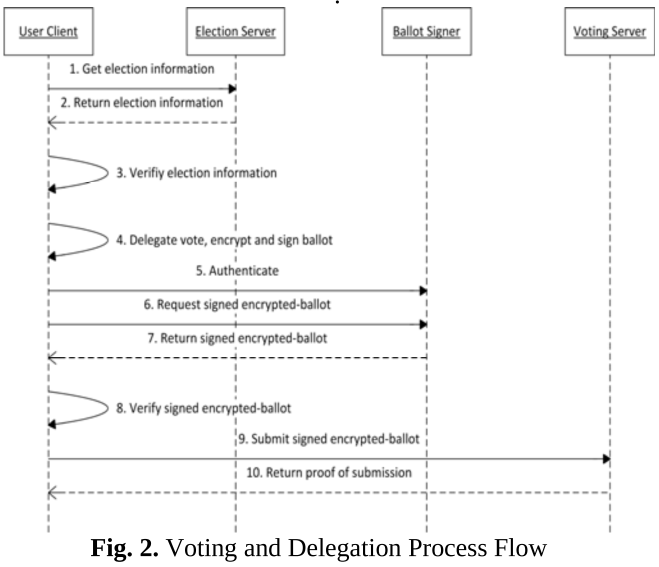
User Client
Election Server
Ballot Signer
Voting Server
1. Get election information
2. Return election information
3. Verifiy election information
4. Delegate vote, encrypt and sign ballot
5. Authenticate
6. Request signed encrypted-ballot
7. Return signed encrypted-ballot
8. Verify signed encrypted-ballot
9. Submit signed encrypted-ballot
10. Return proof of submission

Fig. 2. Voting and Delegation Process Flow

1. Get election information

The *User Client* queries the *Election Server* to get available election information.

2. Return election information

The *User Client* receives election information form the *Election Server*. The received information includes active elections with answer possibilities, the ballot, and an encryption key for the individual election. Additionally, the information on election deadlines or the contents of the election is provided. To prevent manipulation of these data, it is signed by the *Election Server*.

3. Verify election information

The *User Client* verifies the information and the signature received from the *Election Server*.

4. Delegate vote, encrypt and sign ballot

The user selects the desired election and now has the desire to delegate her vote. The *User Client* retrieves the votes of all proxies from a *Voting Server*.

For a *client-based delegation*, the user selects the proxy she wants to delegate her vote and the *User Client* imitates the vote by directly voting for the same answer. For a *server-based delegation*, the user specifies a start and end date of the delegation and specifies the proxy she wants to support.

For both delegation approaches, the user places the vote or delegation locally, and encrypts the filled ballot (encrypted-ballot) using the public encryption key received from the *Election Server*[6]. Finally, the user signs the encrypted-ballot using the signature functionality of her national eID. In our implementation, we relied on the Austrian citizen card, the official eID-system in Austria [13].

5. Authenticate

The user authenticates at the *Ballot Signer* using her national eID to get uniquely identified.

6. Request signed ballot

The *User Client* sends the encrypted and signed ballot to the *Ballot Signer*. The *Ballot Signer* verifies the user's signature. If the signature is valid and the user has not voted for the specific election yet, the *Ballot Signer* will sign the encrypted ballot. If the user has already placed a vote, the *Ballot Signer* will deny the request. Before signing the encrypted vote, the *Ballot Signer* removes the user's signature to further ensure anonymity. Additionally, the *Ballot Signer* generates a rejection code which is also signed by the *Ballot Signer*. This rejection code will not be stored by the *Ballot Signer*.

7. Return signed ballot

The *Ballot Signer* returns the signed ballot and the rejection code to the *User Client*.

8. Verify signed ballot

The *User Client* verifies the signatures of the signed ballot and checks if the signed content is still the same as the original encrypted ballot sent in Step 6.

[6] Note that a non-deterministic encryption scheme is used here.

9. Submit signed encrypted-ballot

The *User Client* submits the encrypted-ballot to a random *Voting Server* that stores the ballot until election end. The *Voting Server* again signs the ballot and sends it back to the *User Client*. This proves the user that the *Voting Server* has accepted the vote. If the vote might vanish from the *Voting Server*, the user knows that the *Voting Server* may be corrupted. The *User Client* can submit the signed encrypted-ballot to further *Voting Servers* to have redundancy in case a single *Voting Server* is corrupted or out of service.

10. Return proof of submission

The *Voting Server* returns the signed and accepted vote to the *User Client* warranting the submission.

2) Rejecting and Revoking of Votes

If the user changes her mind and does not want to support her selected proxy anymore, she should be able to revoke the delegation of her voting power. To reject a vote or revoke a delegation, the *User Client* first authenticates at the *Ballot Signer*. Then the *User Client* sends the signed rejection code (generated and signed by the *Ballot Signer* during the voting process) to the *Ballot Signer*. The *Ballot Signer* verifies its own issued signature of the rejection code and checks if the vote has not already been rejected. If the signature is valid, then the rejection code is published to a rejection list for this election at the *Ballot Signer*. Finally, the *Ballot Signer* allows the user to vote once again.

3) Determine the Election Winner

Every user is able to count the votes and to calculate the results of an election. To achieve this, in a first step all votes (signed encrypted-ballots) for the selected election are downloaded by the *User Client* from all known *Voting Servers* and put on a single list. After that, all digital signatures are checked and votes with invalid signatures are removed from the list. Duplicate votes are also identified and removed from the list.

In a second step, the rejection lists are downloaded by the *User Client* from the *Ballot Signer*. All ballot-IDs on the rejection lists are invalid and are removed from the list.

After the end of the election the *Election Server* publishes its private key for the specific election. All remaining votes are decrypted with this private key and finally counted. Depending on the election model, the *Voting Server* just counts the votes or e.g. weighs the counted votes in a preferential voting system. Our system supports all methods of determining an election winner we have identified as important in Section III.A.

Users stay anonymous with respect to the *Voting Servers* in this determination process as all votes have been placed on the *Voting Servers* encrypted, and signed by the *Ballot Signer* only.

V. Evaluation

In the following, we evaluate our proxy voting system based on the requirements defined in Section III.

A. Functional Requirements

The functional requirements of a proxy voting system are all fulfilled by our proposed solution. All required functions such as voting, delegation of votes, rejecting and revocation of votes, and determination of the election winner can be modeled by our system.

B. Security Requirements

In the following, the security requirements of a proxy voting system are evaluated.

Anonymity: In general, all communication channels between the user and the individual servers are encrypted using SSL/TLS. This ensures that no untrusted third-party might be able to inspect any communization and further disclose a vote and the corresponding user's identity. Additionally, confidentiality of the vote is not only achieved on communication level, but also for the individual entities by encrypting the vote.

In general, all communication channels between the user and the individual servers are encrypted using SSL/TLS. This ensures that no untrusted third-party might be able to inspect any communization and further disclose a vote and the corresponding user's identity. Additionally, confidentiality of the vote is not only achieved on communication level, but also for the individual entities by encrypting the vote.

The vote of a user is encrypted and signed before it is sent to the *Ballot Signer*. Hence, the *Ballot Signer* is not able to read the content of the vote. The *Ballot Signer* authenticates the user, hence it knows the user's identity but it does not store the encrypted vote. Not storing the encrypted vote assures that the user cannot be linked to the encrypted vote after an election ends.

After checking that the user has not voted yet, the *Ballot Signer* removes the signature of the user and signs the vote, before it is sent back to the user. Removal of the citizen's signature ensures anonymity with respect to the *Voting Server*. When the user forwards the vote to a *Voting Server*, the *Voting Server* cannot find out which user the encrypted vote belongs to as the citizen's signature has been previously removed. Hence, the *Voting Server* cannot link the vote to a specific person.

The *Election Server* is not directly involved in the voting process (no votes are transferred to the *Election Server* during the voting process), hence the vote stays always hidden to the *Election Server*.

Secrecy: Secrecy is mainly established by the *Election Server* and the user using public key cryptography. The *Ballot Signer* and the *Voting Servers* cannot read the encrypted vote, because until the election end the private election key is kept secret by the *Election Server*. The *Election Server* is the only server that is able to decrypt the votes. Secrecy with respect to the *Election Server* is assured as it is not directly involved in the voting process.

Integrity: In general, integrity of the system is ensured by using multiple *Voting Servers* and applying digital signatures on the exchanged messages. Votes are stored on multiple *Voting Servers*, whereas all votes are signed by a *Ballot Signer*. Hence, no *Voting Server* can generate or modify any votes. Additionally, a *Voting Server* is not able to invalidate or to reject a vote. Although a *Voting Server* is able to delete a vote, votes are mirrored over multiple servers. If a vote has been deleted by a *Voting Server*, users can prove that the vote was deleted, because they have a signed proof (the encrypted-ballot signed by the accepting *Voting Server*) of having successfully submitted their vote.

The *Election Server* is not involved in the communication process containing the vote; hence it cannot modify, delete, generate, or reject votes. To void the integrity of the system, the *Election Server* could only change the election key or remove the whole election, what can easily be detected by everybody.

The *Ballot Signer* cannot invalidate votes without getting the signed rejection from the user, because the *Ballot Signer* does not store the rejection code. Hence, it also cannot reject votes after the election deadline. The only drawback is that the *Ballot Signer* could generate valid ballots. Utilizing several *Ballot Signers* for each commune and limiting valid votes to the number of eligible voters can minimize this risk.

Authenticity: Ensuring authenticity is the main task of the *Ballot Signer*. In our system, the user authenticates herself to the *Ballot Signer* using her national eID. The use of the Austrian eID ensures unique identification and secure authentication to the *Ballot Signer*.

The *Voting Servers* do not need to authenticate the user, but the validity of the votes is verified by validating the signature provided by the *Ballot Signer*. The *Election Server* is not involved in the communication process containing the vote.

Verifiability: Verifiability is ensured by using digital signatures and separated *Voting Servers* storing the same information. To verify the official result of the election, everybody can download all votes from the *Voting Servers*. After the election ends, the private key of the *Election Server* can be downloaded and the encrypted votes can be decrypted. Then the official result can be also verified by counting the votes by the user themselves locally.

Additionally, each user can check if her own vote is still in the system. Therefore, the user queries the *Voting Server* with the ballot-ID to receive the signed encrypted-ballot. The user compares the received ballot with a local stored copy. If they are unequal, the ballot has vanished and the user can blame the faulty servers, because she has a signed proof of each server that the ballot was accepted.

VI. Conclusions

Liquid democracy and proxy voting represent an interesting approach that bridges the gap between direct and representative voting. Current software systems implementing this approach simplify the whole process and help in a wide range of decision-making processes, such as in e-Government or e-Business. However, due to the lack of security related features, those systems cannot fulfill the requirements of electronic voting systems. Therefore, we have implemented a proxy voting system supporting liquid democracy, which improves current systems in terms of security and privacy.

To reduce the risk of an attack, our system does not store any information on a server that makes reconstruction of votes possible. Separated servers, asymmetric key encryption, and qualified digital signatures are used to make this possible. Unique identification and strong authentication are implemented by using the Austrian citizen card and the associated eID-systems. This prevents users of being able to vote twice. However, due to our distributed architecture and cryptographic functions, we still guarantee anonymity and secrecy when users are casting their votes.

As future work we consider the following aspects as most important: The integration of other European eID systems would be an interesting improvement to facilitate the deployment within the European context. Also, the current version relies on a Desktop-based client application that must be installed by the user. Browser-based clients would significantly improve the usability of the system. However, various security-related problems in relation to a browser-based variant need to be addressed before such an approach can be implemented.

References

[1] Piratenpartei: Liquid Democracy. http://wiki.piratenpartei.de/Liquid_Democracy

[2] Jabbusch, S.: Liquid Democracy in der Piratenpartei, Master Thesis

[3] Lewitzki, M.: Das Internet in Parteiform: Wie segelt die Piratenpartei?, 2010, http://www.regierungsforschung.de/dx/public/article.html?id=*96M*

[4] *Gol*em.de: Piratenbraut mit Doppelleben, http://www.golem.de/news/liquid-feedback-piratenbraut-mit-doppelleben-1303-97952.html

[5] Bieber, C., Lewitzki, M.: Das Kommunikationsmanagement der Piraten. In: Die Piratenpartei. pp 101-124, 2013

[6] Piratenpartei: Liquid Democracy/Votorola. http://wiki.piratenpartei.de/Liquid_Democracy/Votorola, 2012

[7] Zelea.com: User:ThomasvonderElbe GmxDe/Vote mirroring. http://zelea.com/w/User:ThomasvonderElbe_GmxDe/Vote_mirroring #Adhocracy_to_Votorola, 2012

[8] Liquid Democracy e.V.: Über Adhocracy.de - Adhocracy. https://piratenparteinrw.adhocracy.de/_pages/about/uber-adhocracy, 2012

[9] Liquid Feedback: Liquid Feedback – Interactive Democracy. http://liquidfeedback.org/, 2012

[10] Gritzalis, D.: Principles and requirements for a secure e-voting system. In: *Computers & Security*. Vol 21, No 6, pp 539-556, 2002

[11] Rubin, A.: Security Considerations for Remote Electronic Voting over the Internet. In: *Communications Policy and Information Technology: Promises, Problems, Prospects*, 2002

[12] Reynolds, A., Reilly, B., Ellis, A.: Electoral System Design: The New International IDEA Handbook. http://www.idea.int/publications/esd/loader.cfm?csmodule=security/getfile&pageid=10445

[13] Leitold, H., Hollosi, A., Posch, R.: Security Architecture of the Austrian Citizen Card Concept. In: *ACSAC 2002*. pp. 391-402, 2002