A Generic Interface for the Public Bulletin Board Used in UniVote

Severin Hauser, Rolf Haenni
Research Institute for Security in the Information Society
Bern University of Apllied Sciences
CH-2015 Biel, Switzerland
{severin.hauser, rolf.haenni}@bfh.ch

University of Fribourg
CH-1700 Fribourg, Switzerland
severin.hauser@unifr.ch

***Abstract*— To provide verifiability, cryptographic voting protocols usually require a public bulletin board for publishing the election data, so that the data can be read and verified by everyone. The basic requirements on such a board are similar for most protocols, for example that nothing can be changed or deleted from the board. Typically, when it comes to implement a voting protocol in a real system, many additional requirements arise and they can differ from protocol to protocol. This paper shows based on the protocol of UniVote what these requirements might be and what other problems may arise from an operational and organisational point of view. Based on the understanding of these problems, we propose a generic interface for the main board functionalities. This interface offers a flexible way of extending the properties of a public bulletin board to comply with all sorts of additional requirements. We give multiple examples of properties that we identified as desirable for the public bulletin board in UniVote.**

e-voting, bulletin board, verifiablity

I. INTRODUCTION

The concept of a *public bulletin board* (PBB) is an important building block in many cryptographic voting protocols. From a conceptual point of view, a PBB can be regarded as a broadcast channel with memory [1]. Its purpose is to allow the parties involved in a protocol to publish messages, while giving the guarantee that none of these messages can be deleted or modified. A bulletin board that offers at least this *append-only* property is what almost all cryptographic voting protocols have in common. In addition, some protocols require designated board sections for all involved parties [2], while other protocols require that the board rejects messages that are not well-formed [3]. When implementing a PBB, appropriate solutions for such protocol-specific requirements need to be provided in addition to the append-only property.

In a practical application of a PBB in real elections, the situation gets even more complicated. There are numerous operational problems, for example separating the election data when multiple elections run in parallel, opening and closing the electronic ballot box, archiving the election data when the protocol terminates, or ensuring consistent views for all users reading data from the board. This leads to highly specific implementations of public bulletin boards, essentially one for each voting system. In practice, since building an electronic voting system is a very complex and time-consuming process on its own, the development of an appropriate PBB with all desired properties has sometimes been given minor priority. There are several systems with a simplified PBB based on a conventional database system with a public interface. In such systems, the security of the cryptographic protocol gets undermined by an insecure PBB implementation.

A. UniVote

Since 2013, student organizations at various Swiss universities are using *UniVote* to elect their board in an online election [4]. UniVote offers individual and universal verifiability by publishing the election data on a PBB. Its underlying cryptographic protocol is based on mixnets and anonymous authentication [5]. To support the verification process, students have written an independent verification software based on the system specification. This software reads all the data from the board, performs all the cryptographic verifications, and re-computes the election result. The existence of the first verifiable Internet voting system in Switzerland and its usage in real elections has received broad interest.

The first version of UniVote implemented a simplified PBB in form of a database with a public web service interface. The new version of UniVote, which has been used for the first time in September 2015, is based on a more sophisticated PBB. The new board is an independent component that could be used by other

applications. To provide this flexibility, it defines a generic interface for writing and reading the data. Due to this flexibility, it could be adapted easily to the needs of different voting protocols and operational circumstances.

B. Contribution and Paper Overview

The main contribution of this paper is a design proposal for a generic public interface of a flexible public bulletin board (Section III.A). The interface consists of two basic operations, one for posting new messages to the board and one for querying the board's current content. The generic specification of the interface allows these operations to be customized according to the actual requirements of a specific voting protocol. While voting protocols are the primary target application for a PBB implemented according to our design, there are no limitations for applications of our approach in other fields.

The second contribution is a compilation of possible PBB properties, which we think might be useful in different contexts (Section III.B to III.D). In particular, we think that a PBB equipped with these properties can facilitate the implementation of voting protocols. We derive these properties from the voting protocol used in UniVote, which follows the general lines of the most common protocols used in practice (Section II). Our analysis exposes some of the practical problems that may arise with such protocols. To overcome these problems, we introduce additional PBB properties and show how to implement them using the generic interface.

C. Related Work

The idea of publishing the election data on a public bulletin board has a long tradition in the literature of verifiable electronic voting. While almost every existing cryptographic voting protocol uses a PBB as a central communication platform between the parties involved, almost no paper describing such a protocol gives a precise specification of the properties expected from the board. Usually, the existence of an appropriate PBB is just taken for granted, but the PBB itself remains a black box.

Given the importance of the bulletin board concept in electronic voting, only a remarkably small number of specific papers devoted to the problem of specifying and implementing a PBB exists. Peters was one of the first to suggest such a specification and solution [1]. His main focus was on making the bulletin board robust against failures or attacks, using multiple peers and protocols from the multi-party computation literature. In [6], Heather and Lundin made some proposals to ensure the append-only property and to solve the resulting conflicts with the robustness property. Some reports on corresponding implementations have been published later [7, 8]. Another description of a practical PBB implementation is included in the report about the voting system used in the state of Victoria, Australia [9]. In a follow-up paper [10], Culnane and Schneider proposed a robust algorithm for a peered bulletin board and verified its correctness formally.

II. UNIVOTE PROTOCOL OVERVIEW

To illustrate the problems that may arise when a public bulletin board is used in an electronic voting system, we introduce a simplified view on the protocol used in UniVote. We assume that information published on the bulletin board can be read by any party as soon as the bulletin has received the information.

The following parties with different tasks are involved in the voting process:

- *Election Administrator*: Defines the context of the election (title, options, period, list of eligible voters).
- *Bulletin Board Administrator*: Manages the PBB and its server and network infrastructure.
- *Trusted Authority*: Supports the election by participating in some cryptographic computations (e.g. mixing or decrypting encrypted votes) and sharing corresponding keys.
- *Voter*: Casts a ballot during the election.

In the following informal description of the voting protocol, we assume that each voter can write into a designated area of the PBB. We distinguish the three consecutive main phases as described in the protocol:

1) Election Preparation
   a) The election administrator publishes the title of the election, the voting options, and the election period on the public bulletin board.
   b) The election administrator publishes the list of eligible voters on the PBB.
   c) The trusted authorities anonymize the eligible voters
   d) The bulletin board administrator defines a designated area for each anonymized voter.The trusted authorities publish a shared public encryption key on the PBB.

2) Vote Casting
   a) The voters encrypt their votes and publish corresponding ballots in their designated area of the PBB. They receive a confirmation that their ballot has been published.
3) Tallying
   a) The election administrator checks each ballot for validity and publishes the list of valid encrypted votes on the PBB.
   b) The trusted authorities anonymize the valid encrypted votes, perform the decryption in a distributed manner, and publish the election result on the PBB.

In the following subsections, we give a non-exhaustive list of possible PBB-related problems that may arise in UniVote and voting system based on a similar protocol. In each case, we provide some hints about possible solutions.

A. Operational Problems

1) Conflicting Messages

The protocol does not prevent any involved party from submitting a particular message multiple times, possibly with different and conflicting content. One of the simplest solutions to this problem is to declare the first message of a given type and from a given party to be the one that counts. In this case, the board may simply deny all subsequent messages from the same party. On the other hand, if the strategy is to count the last message, then the PBB needs to memorize the order of the published messages based on their time of arrival.

2) Malformed Messages

The protocol requires each message sent to the PBB to have a certain structure and content. If a party sends an incomplete or malformed message—unwillingly or on purpose—, the board needs a strategy for dealing with it. The simplest strategy is to reject the message and deny its publication.

3) Replayed Messages

As messages already published on the PBB can be read by anyone, the protocol does not prevent copying an existing message from someone else and sending it again in the other person's name. If vote updating is permitted in an election, this can be exploited for cancelling an updated vote by simply submitting the first vote a second time. To prevent this, the PBB must be able to verify the freshness of the message. Again, there are multiple strategies for avoiding such problems, for example by making each post dependent on its predecessors.

4) Early and Late Messages

In the first step of the protocol, the election administration specifies the election period. Only ballots posted during this period are accepted in the final tally. For ballots arriving too early or too late, the board needs a strategy of handling them. The simplest general solution is to reject all messages not arriving within the bounds of a specified time slot. Since rejecting messages is a delicate issue, especially in case of ballots in an election, they could also be published together with a timestamp. Generating unforgeable and accurate timestamps is another problem on its own.

5) Board Flooding

Depending on the board's strategy of dealing with conflicting, malformed, replayed, and early/late messages, the involved parties may be allowed to post arbitrarily many messages. This can be exploited by someone who wants to sabotage the election. By sending a huge amount of messages with unrelated content, the board can be flooded with messages until its capacity is reached. This problem gets even worse, if the PBB accepts messages from users outside the protocol.

6) Secure Authentication

The protocol states that bulletin board administrator creates designated areas for voters to publish their ballots. This implies that the board is able to authenticate the author of every posted message and to reject messages from unauthorized parties. The necessary infrastructure for providing secure authentication is not defined by the protocol, but needs to be specified carefully in an actual implementation.

7) Undeniable Receipts

When a voter publishes a ballot, the protocol requires the PBB to respond with a confirmation. The goal is to supply voters with a proof that their ballot has been published. The PBB must therefore be able to create receipts for published messages that are not deniable. In an actual implementation, the details of such undeniable receipts need to be specified.

8) Consistent Views

To verify the outcome of an election, the entire election data needs to be retrieved from the PBB. Clearly, the verifiers will only come to a consistent conclusion, if they all obtain exactly the same view of the board. The process of querying the board must therefore include measures that guarantee consistent views for all users and under all circumstances.

B. Organisational Problems

1) Extending the Election Period

Due unexpected circumstances like power outages, network or server problems, or misinformation, it might be necessary to readjust the election period during an election. The protocol does not explicitly include a procedure for dealing with such an exceptional situation. In a proper solution, the election administrator must announce the extended period on the PBB. This implies that the above-mentioned strategy of dealing with late messages must be adjusted accordingly.

2) Multiple Elections

In a productive environment, setting up the PBB for every election is not very practical. The board should therefore be designed to support multiple elections, for example by providing designated areas for different elections, which are strictly separated.

3) Simultaneous Elections

With a PBB supporting multiple elections, it might happen that some elections run at the same time, possibly with different administrators, trusted authorities, or voters. Handling multiple elections simultaneously is another desirable property for a practical PBB, which needs to be considered in its design.

4) Closing an Election

At the end of an election, when the final result has been published and accepted, adding further messages to the election data should be prohibited. The PBB should therefore provide a mechanism for closing an election by locking up the final state of the election data and by making this visible to the public.

5) Archiving an Election

In the strict sense of the word, an append-only PBB must never allow the deletion of some of its contents. On the other hand, keeping the election data from all the elections in the past may not really be necessary or meaningful. The PBB should therefore allow the removal of the data of an entire election as soon as the public has verified and accepted the final election result. To archive the data for future use, it can possibly be moved to another place that does not the debit the capacity and resources of the bulletin board.

III. GENERIC INTERFACE AND PROPERTIES

To address the aforementioned problems in a uniform way, we require a flexible formal definition of what a PBB actually is and how it works. Clearly, its main purpose is to store the set of published messages and allow users to retrieve them. As additional board properties may require some kind of metadata to be added to each message, let $p = (m, \alpha, \beta)$ be a container for a message $m \in M$ and its metadata. Furthermore, we differentiate the metadata included in such a *post* p by its origin as it can be added either by the author of the message (the user) or by the board, and use different symbols $\alpha \in A$ and $\beta \in B$, respectively. We will model α and β as lists of *attributes* (called *user attributes* and *board attributes*, respectively). For a set of indices I, we write α_I and β_I for selecting corresponding sub-lists of attributes from α and β.

In this extended model, the purpose of the PBB is to store a set of posts rather than a set of messages. We use P_t to denote this set at some given point in time t. This set represents the board's internal state (we will later omit the index t and simply write P for the board's internal state). Note the important subtlety of defining P_t as a set and not as a list. The *append-only* property, which according to our interpretation means that nothing can be deleted from the board, can then be defined by $P_t \subseteq P_{t'}$ for all $t' \geq t$. This is the minimal property any PBB should satisfy. If knowing the order in which the messages have been posted to the board is required by an application, then corresponding board attributes have to be added to the post when it arrives on the board (see Section III.C).

In the remainder of this section, we first introduce the generic interface consisting of two basic operations for posting and retrieving messages. We then define multiple PBB properties which may help solving the problems identified in the previous section. The solution proposed for every property is realized within the boundaries of the generic interface by corresponding user and board attributes. With distinguish between properties for structuring the board content, keeping track of the board's history, and ensuring the authenticity and integrity of the posts.

A. Basic Operations

We consider two basic operations that any PBB needs to provide in its public interface. We call them $Post$ and Get. Our goal is to design them in a generic way so that we can change the properties of the board without changing the signature of the operations. In principle, a PBB could support more operations such as $Update$ or $Delete$, but these are in contradiction with the append-only property. Non-public operations for setting up and managing the board are not part of the public interface.

1) Post

When publishing a message $m \in M$, we said that the user will also provide some user attributes $\alpha \in A$. Upon receiving m and α, the PBB might do some checks to validate the post. In case a check fails, an error message $\perp$ is returned and the procedure aborts. Otherwise, some board attributes $\beta \in B$ are generated and the post $p = (m, \alpha, \beta) \in M \times A \times B$ is formed. We model both $\alpha = (\alpha_1, \ldots, \alpha_u)$ and $\beta = (\beta_1, \ldots, \beta_v)$ as corresponding lists of values $\alpha_i \in A_i$ and $\beta_i \in B_i$ without further specifying the sets A_i and B_i. Note that not necessarily all combinations of attributes might be permitted, which is why we define $A \subseteq A_1 \times \cdots \times A_u$ and $B \subseteq B_1 \times \cdots \times B_v$ as subsets of the corresponding Cartesian products.

To conclude the procedure of posting something to the PBB, the board's current state is updated to $P \leftarrow P \cup \{p\}$ and β is returned to the user. Note that by receiving β as a response of posting (m, α), the user gets in possession of the full post p. The following signature summarizes the $Post$ operation:

$$\beta \vee \perp \leftarrow Post(m, \alpha).$$

2) Get

To obtain the simplest possible operation for retrieving the board's current content, we could define it as $P \leftarrow Get(\ )$. The users would then always obtain every single post contained in P, even if only some particular posts are of interest. In a productive environment, where P could grow into a very large set, this solution might not be very practical. Therefore, we let the user define a *query* $Q \subseteq M \times A \times B$, which is applied as a filter to the elements of P. Therefore, the result of the query is the set $R = P \cap Q$. Note that an unconstrained query $Q = M \times A \times B$ results in returning the full set P as above.

In addition to returning R, the board might also produce and return some metadata $\gamma \in C$ about the result of the query. As above, the metadata is modelled as a list $\gamma = (\gamma_1, \ldots, \gamma_w)$ of attributes $\gamma_i \in C_i$ without further specifying the sets C_i. We call them *result attributes*. The following signature summarizes the Get operation:

$$R, \gamma \leftarrow Get(Q).$$

For better readability, we sometimes write P_Q for the resulting subset of posts $R = P \cap Q$. For a query restricting only the i-th user attribute to a single value $\alpha_i \in A_i$ or to a subset of values $A_i \subseteq A_i$, we use the simplified notation $Q = \langle \alpha_i \rangle$ and $Q = \langle A_i \rangle$, respectively. Similarly, we write $\langle \alpha_i, \alpha_j \rangle = \langle \alpha_i \rangle \cap \langle \alpha_j \rangle$ or $\langle A_i, A_j \rangle = \langle A_i \rangle \cap \langle A_j \rangle$ for restrictions on multiple user attributes i and j. The same notational convention can be applied to board attributes or to mixed restrictions on user and board attributes.

B. Structuring the Board Content

1) Property 1: Sectioned

A public bulletin board is called *sectioned*, if it consists of multiple equally shaped sections. The goal of a sectioned bulletin board is to separate unrelated messages into logically independent units. Let S denote the set of available sections. To enable the dispatching of an incoming post into the right section, the author must provide the section $s \in S$ as a user attribute. A post containing an invalid section $s \notin S$ is rejected by the board. In an election application, it would be natural to define individual sections for the data of each election. Therefore, this property addresses the problem of using a PBB for multiple, possibly simultaneous elections (see Section II.B).

2) Property 2: Grouped

In a *grouped* bulletin board, messages are organized into groups. Typically, messages contained in the same group are similar in shape and content. Let G be the set of available groups. When posting a message, the author must indicate the group $g \in G$ to which the message belongs as a user attribute. A post containing an invalid group $g \notin G$ is rejected by the board. Note that groups are independent of sections, i.e., every section in a sectioned board consists of the same set of groups G. Figure 1 shows an example of a board with three sections and three groups. In an election application, we would define individual groups for every different post in the protocol, for example for the list of candidates or the ballots. Introducing

groups in a PBB used for elections does not directly address one of the problems listed in Section II, but it is a prerequisite for the following property.

3) Property 3: Typed

A grouped bulletin board is called *typed*, if each group $g \in G$ defines its own set $M_g \subseteq M$ of valid messages. M_g is called *type* of g. In a typed board, an incoming message m for group g is accepted only if $m \in M_g$, while all other messages $m \notin M_g$ are rejected. The example in Fig 1 shows a typed board with different types of messages for each group, for example $M_{Group1} = \{0, \dots, 9\}^5$, $M_{Group2} = \{A, \dots, Z\}$, and $M_{Group3} = \{0,1\}^8$. By using a typed PBB for an election, we address the problem of malformed messages (see Section II.A).

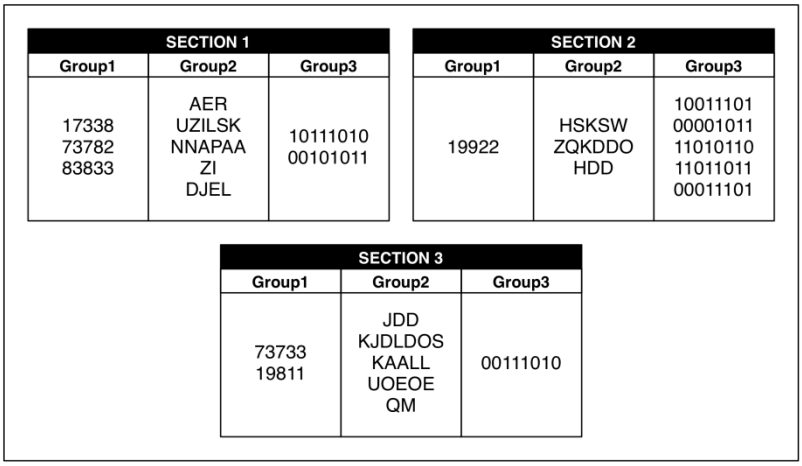

Figure 1: Example of structured bulletin board with three sections, three groups, corresponding types and some messages.

C. History of Board Content

1) Property 4: Ordered

In an *ordered* bulletin board, the posts $P_{\langle s \rangle}$ published for a given section $s \in S$ are ordered according to their time of arrival.[1] This can be achieved by adding a sequence number $i \in N$ to each post, for example $i = |P_{\langle s \rangle}|$ to obtain consecutive numbers 0,1,2, ... for each section. In our generic setting, the PBB includes this number as a board attribute in β. Note that keeping track of the general message order over all board sections is not mandatory, since we consider sections as logically independent units. In an election system, knowing the message order is important for solving the problem of conflicting messages, for example in case the voter submits multiple ballots (see Section II.A).

2) Property 5: Chronological

A bulletin board is called *chronological*, if a timestamp $t \in T$ indicating the exact time a arrival is added to every incoming post. The PBB is responsible for generating accurate timestamps and adding them as board attributes to β. Note we cannot exclude that multiple posts receive identical timestamps, especially in case of a coarse time unit. A chronological PBB is therefore not automatically ordered. In an election system, attaching timestamps to ballots is important to decide whether they have been received within the election period, or more generally to solve the problem of early or late messages (see Section II.A).

3) Property 6: Interlinked

An ordered bulletin board is called *interlinked*, if every post in every section depends on all its predecessors in that section. More formally, let $p_i \in P_{\langle s \rangle}$ denote the post with sequence number i in a given section $s \in S$ and $P_{\langle s,\{0,\dots,i-1\}\rangle} = \{p_0, \dots, p_{i-1}\}$ the set of predecessors of p_i in $P_{\langle s \rangle}$. Interlinking p_i with all its predecessors can be achieved by applying some function H (typically a hash function) to $P_{\langle s,\{0,\dots,i-1\}\rangle}$. Another solution is to apply H only to the previous post p_{i-1}, which then creates indirect links to all predecessors of p_{i-1}. Such a construction is sometimes called *hash chain*. In our generic setting, the resulting value $H_i = H(p_{i-1})$ can be included in $p_i = (m, \alpha, \beta)$ either as a user attribute in α or as a board attribute in β (an initial value H_0 is added to p_0). The first option corresponds to the construction proposed in [6] for achieving the append-only property. It implies that no pair (m, α) will appear more than once on the board and thus eliminates the

1An unsectioned PBB can always be considered as a board with a single section

problem of replayed messages (see Section II.A). However, due to coordination problems between users posting messages simultaneously, it is rather complicated to implement properly. In combination with other properties from the next subsection, an interlinked PBB also helps addressing the problems of undeniable receipts and consistent views.

D. Authentication and Integrity

1) Property 7: Access-Controlled

A bulletin board is called *access-controlled*, if it provides an access-control mechanism that authenticates the author of a message and rejects the message if the user is not authorized. To enable the PBB doing this check, we assume that a set K of public signature keys—one for each authorized user—is known to the board at every moment. This set is either *static* or *dynamic*. In the static case, K is publicly known and can not be changed, whereas in the dynamic case, $K = K(P_t, \alpha, \beta)$ is defined implicitly by a publicly known function K, which depends on the current board state P_t and the attributes included in the incoming post $p = (m, \alpha, \beta)$. The three arguments of K are optional, i.e., not all of them are relevant in every case. For p to be accepted by the board, the user's public key pk and a signature $S = Sign_{sk}(m, \alpha_I)$ must be included as user attributes in α (we use α_I to denote the list of user attributes different from pk and S). The board can then perform the checks $pk \in K$ and $Verify_{pk}(m, \alpha_I; S)$ to decide whether p stems from an authorized user or not.

In electronic voting, our proposal of a dynamic set K may serve multiple purposes. For example, we can define a function K that allows the election administration or the trusted authorities to post exactly one message of a given type. Similarly, in a system that prohibits vote updating, we can give voters the right to submit exactly one ballot. For this, K must depended on P_t (to check if a message of the same type has been posted earlier by the same author) and on α (which contains the author's public key). This mechanism is therefore a solution for the problem of conflicting messages. Together with other measures against malformed or replayed messages, it also helps avoiding board flooding attacks (see Section II.A).

Another possible application of a dynamic set K is to restrict the voter's right to submit ballots to the election period. In a chronological PBB, every post contains a timestamp in the list of board attributes, which implies that in this case K must depend on β. Using such quantitative and temporal restrictions on the board's access rights, we can properly implement the process of closing an election (see Section II.B). As soon as all access rights have been expended or have expired, the board's content reaches a final state. In a sectioned PBB, a final state can be reached for each individual section. This is a precondition for archiving the data of an election after some time.

2) Property 8: Certified Publishing

A bulletin board offers *certified publishing* [6], if the board attests any response returned to a user with a digital signature. We distinguish between two sub-properties *certified posting* and *certified reading*, depending on the operation. Upon receiving (m, α) from a user calling the $Post$ operation, the board generates a signature $S_{Post} = Sign(m, \alpha, \beta_I)$ and adds S_{Post} as a board attribute to β. With β_I we denote the list of board attributes different from S_{Post}. Similarly, upon responding a user's query Q with the result R, the board generates a signature $S_{Get} = Sign(Q, R, \gamma_I)$ and adds S_{Get} together with a timestamp t as result attributes to γ. Again, γ_I denotes the list of result attributes different from S_{Get} (but including t). Recall that returning β and γ to the user is already part of the $Post$ and Get operations in the generic interface.

In both cases, the user can check the validity of the signature using the board's public key. Note that in an interlinked PBB, the signature S_{Post} is not only a receipt for the publication of the message, but also a commitment to the current content of the board. Similarly, each signature S_{Get} is a commitment of the board to its content at time t. By issuing such commitments with each accepted post and for each query, the board guarantees the consistency of its history and therefore the integrity of the stored data. In an election system, this is a precondition for offering consistent views of the election data to every verifier (see Section II.A).

E. Putting Everything Together

To conclude this section, let us consider a bulletin board satisfying all the properties described above (the variant in which the hash chain is generated by the board). To post a message $m \in M_g$ to the board, the user must provide a list of user attributes $\alpha = [s, g, pk, S]$ containing a section $s \in S$, a group $g \in G$, the user's public key $pk \in K$, and a signature $S = Sign_{sk}(m, [s, g])$ generated using the user's secret key sk. If the

post is accepted, the board responds with a list of board attributes $\beta = [i, t, H_i, S_{Post}]$ containing a sequence number $i \in N$, a timestamp $t \in T$, a hash value $H_i = H(p_{i-1})$, and a signature $S_{Post} = Sign(m, \alpha, [i, t, H_i])$:

$$[i, t, H_i, S_{Post}] \vee \perp \leftarrow Post(m, [s, g, pk, S])$$

If a query Q is sent to the bulletin board, it responses with the result R and a list of result attributes $\gamma = [t, S_{Get}]$ containing a timestamp $t \in T$ and a signature $S_{Get} = Sign(Q, R, [t])$:

$$R, [t, S_{Get}] \leftarrow Get(Q).$$

IV. Conclusion

In this paper, we discussed several important practical aspects to consider when implementing a public bulletin board. We identified a number of operational and organizational problems of using a PBB in an election system like UniVote, and we proposed multiple board properties to address them. The solutions suggested for each property all support the same generic interface, which we defined as a common ground for making PBB implementations more flexible and adaptable to different needs.

An important aspect that remains for future work is robustness. Techniques like the ones proposed in [1, 6, 10] need to be represented as properties.

Acknowledgment

We thank the anonymous reviewers for their thorough reviews and appreciate their comments and suggestions. This research has been supported by the Hasler Foundation (Project No. 14028).

References

[1] Peters, R. A. 2005. "A Secure Bulletin Board." Master's thesis, Department of Mathematics; Computing Science, Technische Universiteit Eindhoven, The Netherlands.

[2] Cramer, R., R. Gennaro, and B. Schoenmakers. 1997. "A Secure and Optimally Efficient Multi-Authority Election Scheme." *European Transactions on Telecommunications* 8 (5): 481–490.

[3] Haenni, R., and R. E. Koenig. 2013. "A Generic Approach to Prevent Board Flooding Attacks in Coercion-Resistant Electronic Voting Schemes." *Computers & Security* 33: 59–69.

[4] Dubuis, E., S. Fischli, R. Haenni, S. Hauser, R. E. Koenig, P. Locher, J. Ritter, and P. von Bergen. 2013. "Verifizierbare Internet-Wahlen an Schweizer Hochschulen Mit UniVote." In *INFORMATIK 2013, 43. Jahrestagung Der Gesellschaft Für Informatik*, edited by M. Horbach, 767–788. LNI P-220. Koblenz, Germany

[5] Haenni, R., and O. Spycher. 2011. "Secure Internet Voting on Limited Devices with Anonymized DSA Public Keys." In *EVT/WOTE'11, Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*, edited by H. Shacham and V. Teague. San Francisco, USA.

[6] Heather, J., and D. Lundin. 2008. "The Append-Only Web Bulletin Board." In *FAST'08, 5th International Workshop on Formal Aspects in Security and Trust*, edited by P. Degano, J. Guttman, and F. Martinelli, 242–256. LNCS 5491. Malaga, Spain.

[7] Krummenacher, R. 2010. "Implementation of a Web Bulletin Board for E-Voting Applications." Master's thesis, Switzerland: Hochschule für Technik Rapperswil (HSR).

[8] Beuchat, J. 2012. "Append-Only Web Bulletin Board." Master's thesis, Biel, Switzerland: Bern University of Applied Sciences.

[9] Burton, C., C. Culnane, J. Heather, T. Peacock, P. Y. A. Ryan, S. Schneider, S. Srinivasan, V. Teague, R. Wen, and Z. Xia. 2012. "A Supervised Verifiable Voting Protocol for the Victorian Electoral Commission." In *EVOTE'12, 5th International Workshop on Electronic Voting*, edited by M. Kripp, M. Volkamer, and R. Grimm, 81–94. Lecture Notes in Informatics P-205. Bregenz, Austria.

[10] Culnane, C., and S. Schneider. 2014. "A Peered Bulletin Board for Robust Use in Verifiable Voting Systems." *Computing Research Repository* arXiv:1401.4151.