Introducing Verifiability in the POLYAS Remote Electronic Voting System

Maina M. Olembo, Patrick Schmidt and Melanie Volkamer
CASED, TU Darmstadt,
Mornewegstraße 32
64293 Darmstadt, Germany
{maina.olembo, patrick.schmidt, melanie.volkamer}@cased.de

***Abstract*—Remote electronic voting continues to attract attention. A greater number of election officials are opting to enable a remote electronic voting channel. More and more scientific papers have been published introducing or improving existing remote electronic voting protocols. However, while the scientific papers focus on different aspects of verifiability, most of the systems in use do not provide verifiability. This gap is closed in this paper by extending a widely used remote electronic voting system, the POLYAS system, to provide verifiability. This approach has been tested in the 2010 election of the German Society for Computer Scientists and will be applied in future elections.**

***Index Terms*—Security, Communication System Security, Data Security, Information Security, Cryptography**

I. Introduction

POLYAS is a remote electronic voting system that has been in use since 1996 in various national and international elections including those for the DFG - Deutsche Forschungsgesellschaft (German Science Foundation) -, the Initiative D21 Association, the Swiss Life Group, which is an insurance company in Switzerland, and both Finnish and German youth elections. Recently, it was used to enable remote electronic voting for the first legally binding university election at the Friedrich Schiller University in Jena. The most popular example has been the annual elections of the GI - Gesellschaft für Informatik (German Society for Computer Scientists) -, where it has been used parallel to postal voting since 2004 (in 2010 for example, 3193 members cast an electronic vote and 51 a mail vote). POLYAS has so far successfully handled all these elections. It is estimated that as of 2010, about one million legally binding votes have been cast using this voting system [1].

In 2008, the BSI - Bundesamt für Sicherheit in der Informationstechnik (German Federal Office for Information Security) -, certified and released the Common Criteria Protection Profile defining a 'Basic set of requirements for Online Voting Products' [2]. Based on this protection profile, POLYAS is currently the first remote electronic voting system that is evaluated according to this international Common Criteria (ISO/IEC 15408) evaluation standard [3]. This evaluation covers only basic requirements for remote electronic voting systems and does not consider verifiability as one of the requirements to be fulfilled. Consequently, verifiability is not evaluated by the Common Criteria evaluators. For the type of elections executed with POLYAS so far, that is with a low public profile and low security risk, this is acceptable. First, because the application note of [2] states that 'this Protection Profile is sufficient to securely implement some kinds of elections in associations, for boards and bodies such as at universities, within the scope of education and research, and in particular other non-political elections with low attack potential'. Second, a paper by Buchmann and Roßnagel [4] states that the German Federal Court decision [5] from 2009 does not hold for all elections, but mainly for German parliamentary elections.

However, if POLYAS as one of the widely used remote electronic voting systems should be used for elections bearing a higher public profile and therefore a higher security risk, in particular for parliamentary elections in Germany (e.g. as an additional channel to mail voting), it has to incorporate verifiability as required by the court decision [5][1]. The goal of this paper therefore is to present an extension to the POLYAS system in order to provide verifiability. We also discuss the application of this extension in light of the GI 2010 election, share our experiences and lessons learnt including modifications of the voting procedure for the 2011 election.

The remaining part of this paper is structured as follows: In Section II we present a description of the POLYAS system; in Section III we execute a security analysis of the system. Section IV proposes verifiability approaches and afterwards, in Section V, the application of these approaches in the GI 2010 elections and the plans for future GI elections are discussed. Section VI concludes the paper.

II. System Description

The electronic voting system POLYAS can be classified in several ways. According to the classification presented in [6] POLYAS is described as a remote electronic voting system that uses:

This work was supported by CASED (www.cased.de) and Micromata (www.micromata.de).

[1]We do not state that POLYAS is ready for parliamentary elections once it implements verifiability. This is a necessary but not a sufficient condition.

This work has been published in the Proceedings of the Sixth International Conference on Availability, Reliability and Security 2011. IEEE 2011 ISBN 978-1-4577-0979-1. DOI: http://doi.ieeecomputersociety.org/10.1109/ARES.2011.26

- A secret-based technique for voter authentication. The voter ID is known a priori to the voter, such as a membership number or ID number. The authentication token is a voting TAN which is sent to the voter by postal mail and is used for one election only. Note, the integration of smart cards for voter authentication is possible, but as it is irrelevant for verifiability purposes, we do not discuss it in this paper.
- The "separation of duty" principle to ensure electoral secrecy in the voting phase.
- A web browser without JavaScript (thin client) as the voter interface.

In the following subsections we first introduce the components involved and their responsibilities, describe the voting procedures and protocols, and then provide some details about the implementation relevant for verifiability considerations.

A. Components and Responsibilities

POLYAS comprises a Vote Casting Interface (VCI) and three servers, namely, the Electoral Registry Server (ERS), the Validation Server (VS) and the Ballot Box Server (BBS). An off-line Tallying Component (TC) is used to tally votes. Another component, the Committee Tool (CT), enables election officials to, for example, start and stop the election. Note, the CT is not discussed further as it is irrelevant for verifiability purposes. The components and their interaction are illustrated in Fig. 1.

The VCI is a web interface displayed to the voter (developed using HTML code) that can be accessed by different web browsers, including Lynx, a text-based browser. It facilitates voter authentication and casting of votes. The ERS authenticates voters and checks their eligibility to participate in an election; the VS controls the ERS by additional authentication of voting material from voters; and the BBS stores the cast votes in an encrypted form. The TC loads the encrypted votes, decrypts them, and computes the election result.

B. Voting Procedures and Protocols

This subsection describes the voting procedures and protocols of the three phases of interest, namely pre-voting, voting and post-voting. Note that we abstract details of the protocol and present those of interest for verifiability.

Pre-Voting Phase: In this phase, a series of six steps is carried out by election officials.

First, voting TANs are generated. Two lists are then produced: a hashed list and an encrypted list of voting TANs. The former list contains, in addition, voter ID information and is placed in the electoral register. The latter list is sent with the voters' names and postal addresses to a service provider which prints the election material. This service provider decrypts the TANs using its private key and allocates a decrypted version to each piece of election material. These are then dispatched to the voters using the postal address information provided. Each TAN is hidden in the form of a scratch field such that the voter has to reveal it.

In the *second step*, key pairs are generated for the three servers and the TC, namely:

- HTTPS key pairs for all three servers,
- a signature key pair for the BBS, and
- a database key pair for the TC.

The HTTPS public keys for the ERS and BBS are published on the election web page and the corresponding fingerprints are printed on the election material. In addition, public keys are distributed to the different components. The components contain the following keys (see also Fig. 2):

- The ERS has an HTTPS key pair for communication with voters and other components. It also stores the public HTTPS keys for the other two servers and the public signature key of BBS.
- The VS contains an HTTPS key pair for communication with other components. In addition, it stores the public HTTPS keys for the other two servers.
- The BBS has an HTTPS key pair for communication with voters and other components and the public HTTPS keys for both ERS and VS. It also stores the private signature key of the BBS and the public database key from the TC, which is used for vote encryption.
- The TC contains the private key of its database key pair, which it uses to decrypt votes at the end of an election. This private key is stored encrypted and the decryption is protected by two passphrases, each known to only one (independent) election official.

In the *third step*, the electoral register is installed in the ERS. In addition, the list of eligible TANs (not hashed) is loaded in the database of the VS and it is verified that the ballot box database at the BBS is empty. *Next*, the ballots for a particular

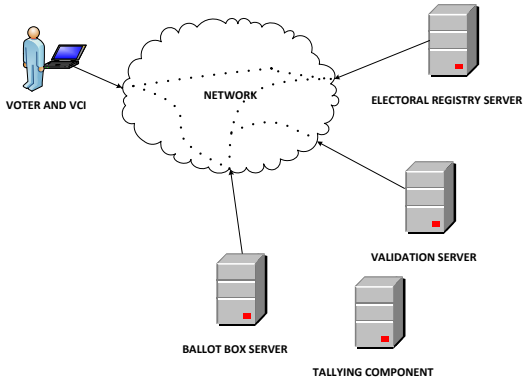

Figure 1. Interaction of POLYAS components

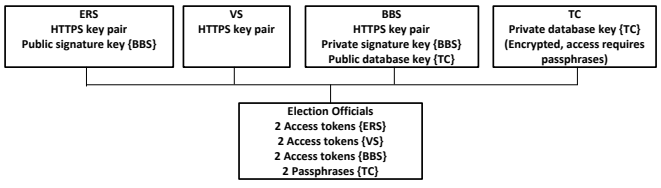

Figure 2. Components, election officials, and assigned keys

election (including name of candidates and title of election) and voting rules (including for invalid votes) are set up on the BBS. In the *fifth step*, two access tokens are generated for each server. Both must be entered to remotely access the corresponding server. These six tokens are distributed to six independent election officials. In the *final step*, the three servers are configured and security software, such as anti-virus software, is installed. POLYAS software is then installed and the election is started by the election officials.

Voting Phase: In Fig. 3 the steps involved in the voting phase are illustrated. Communication between the different components is secured using Secure Socket Layer (SSL). The secured SSL communication is illustrated as: between the voter and ERS, SSL_1; between ERS and VS, SSL_2; between VS and BBS, SSL_3; between the voter and BBS, SSL_4; and between ERS and BBS, SSL_5.

Assuming the communication between the components is ensured by SSL, the voting protocol can be described in the following way:

The voter visits the election web page and, ideally, verifies the SSL certificate of the ERS. He then presents his *voter ID* and *voting TAN* (**1**). The ERS verifies the voter's eligibility in the electoral register. If the voter is eligible, the ERS sends the *TAN* to the VS (**2**). The VS verifies the voter's eligibility by checking if its database contains this *TAN*. If it does, the VS generates a random voting token *T*. This voting token is generated only if both ERS and VS decide that the voter is eligible. Note also that if the VS has been presented with the same TAN previously, it will not generate a new token; rather it will send the same token back to the ERS (but not to the BBS).

After the VS has generated a new token *T*, it sends it to the BBS (**3**). The BBS temporarily stores this voting token and sends back an acknowledgement (**4**). After receiving this acknowledgement, the VS sends the same voting token *T* to the ERS (**5**) which temporarily stores it and sends an acknowledgement to the VS (**6**). In order to prevent double voting, after receiving this acknowledgement, the VS marks the hashed TAN as invalid, and sends a confirmation to ERS (**7**).

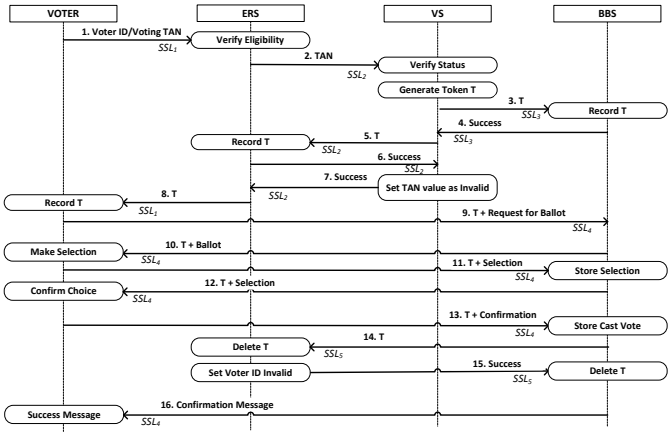

Figure 3. The POLYAS voting protocol

The ERS sends *T* to the voter (**8**) who is then automatically forwarded to the BBS to make a request (containing *T*) for a ballot (**9**). Ideally, the voter verifies the SSL certificate of the HTTPS connection with the BBS.

The BBS verifies whether *T* is a valid voting token. The BBS considers *T* to be valid if it has been received from the VS and has not yet been deleted (as this would mean a vote has already been cast). If this is the case, it sends the *ballot* to the voter (**10**). The ballot is displayed to the voter in his browser and he makes a selection from the options available. The token *T* and voter's *selection* are sent to the BBS (**11**). If the voting token *T* is valid the BBS encrypts and stores the selection in the ballot box database together with the voting token *T*. This entry is marked as 'selection'. It then sends back the voting token and the stored selection (in plain text using only HTTPS encryption) plus information whether it is a valid or invalid vote (**12**). The selection is then displayed again to the voter. The steps herein can be repeated if the voter wants to change his selection, prior to casting his final vote. Finally, the voter casts his vote by confirming this in the browser. A corresponding confirmation message including the voting token *T* is sent to the BBS (**13**). The BBS verifies whether *T* is still valid. If this is the case the BBS changes the status of the entry in the database belonging to voting token *T* from 'selection' to 'cast vote'.

After this, the BBS sends the voting token *T* to the ERS (**14**) which deletes the token it temporarily recorded. In addition, it sets the corresponding voter ID as invalid. Next it sends an acknowledgement message to the BBS (**15**) which then deletes the corresponding voting token from the database and any other temporarily stored copies. By deleting voting tokens voter privacy is maintained as any link between the voter and his cast vote are then removed. The voter will receive a confirmation message of a successfully cast vote (**16**) and this marks the end of the voting phase[2].

Cast votes are stored in a randomized order in the ballot box database, in blocks of thirty. These blocks are introduced to implement a mechanism that would allow verification if problems during the post-voting phase are detected. This mechanism works in the following way: A hash chain is created by applying a hash function to a block of thirty encrypted votes. Thus, as soon as the BBS has received the first thirty votes, it concatenates these encrypted votes, attaches an initial hash value, computes the hash, and signs the output using the private signature key. The output and the signed version are sent to the ERS. The signature is verified and, if valid, the message is stored in a separate database at the ERS. An acknowledgement message is sent back to the BBS. The next block of thirty votes is attached to this hashed output and the hash function applied again. This process is repeated for all available votes. At the end of the election the last block of votes may contain less than thirty votes. In such a case, the

[2]VCI screenshots are available at https://www.gi.de/wahlen2010/.

votes are not included in a hash chain. The steps to create a hash chain are illustrated in Fig. 4.

Post-Voting Phase: At the end of the election period, the election officials close the election by using the different access tokens to get access to the three different components. Election officials download the ballot box database from the BBS, the electoral register (including the status who has cast a vote) from the ERS, and the database from the VS. The ballot box database is loaded in TC. The two passphrases are entered and tallying is carried out. The results are displayed and printed out. Note that only those votes where the status is set to 'cast' are taken into account, and not where the status is set to 'selection'. The cast (decrypted) votes are printed out and stored to enable manual verification, should disputes arise. The number of cast votes according to the VS and the ERS are deduced from the corresponding databases and compared with the number of votes in the ballot box database. In addition, the database content of the three servers is archived on a CD.

C. Implementation Details

POLYAS is implemented in Java and integrates the Bouncy Castle Cryptographic APIs. In this subsection, we present only those implementation details required for verifiability (see Section IV).

- **Keystore:** The private database key from the TC is stored in a Java keystore. According to the Java implementation one needs one passphrase to access the keystore and another passphrase to access the private key. Therefore both passphrases need to be entered in order to access the private key, decrypt the votes in the database and tally them.
- **Encryption of votes:** A hybrid encryption scheme is used to encrypt votes based on RSA-1024 and AES-256. For each received selection, the BBS generates a temporary (symmetric) key k_{AES} which is used to encrypt the vote. k_{AES} is then encrypted with RSA using the public database key from the TC.
- **Hash function:** SHA-256 is used to generate the hash chain. An initial hash value (for the first block of thirty votes) is hard coded in the source code.
- **Ballot Box Database:** A database entry in the BBS contains the encrypted vote, the block identifier, the order of the vote in the block (1 to 30), and the status (refer to Table I).

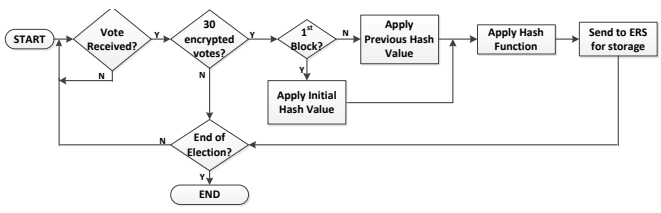

Figure 4. Flowchart illustrating creation of hash chain

Table I
BBS DATABASE CONTENTS

Encrypted vote	Block Number	Order in Block	Status
enc($vote_{24}$)	1	1	cast
⋮	⋮	⋮	⋮
enc($vote_{13}$)	1	30	cast

III. SECURITY ANALYSIS

A variety of security requirements for remote electronic voting schemes has been proposed such as in [6], [7], [8], [9], and [10]. In this analysis, however, we only address integrity of the election results which is relevant to verifiability. According to [6] integrity of election results can be violated in four different situations: at the voter's PC, on the network, at the voting servers, and/or at the tallying component.

A. Integrity Requirements

We will analyse if POLYAS is susceptible to the following threats according to those listed in [2]:

- T1 Malicious software on the voter's PC is able to modify single votes undetected and thereby manipulate the election result.
- T2 A manipulated VCI is able to get knowledge of the voting login credentials (here: voting TAN and voter ID) and cast a vote on behalf of the voter.
- T3 A manipulated VCI is able to modify the cast vote undetected before sending it to the voting server.
- T4 A single manipulated voting system component is able to add or delete voters in the electoral register undetected.
- T5 A single manipulated voting system component is able to add, delete or modify stored votes in the ballot box.
- T6 An arbitrary attacker is able to modify votes on the network undetected.
- T7 The TC is able to output a result that does not correspond to the votes in the ballot box without being detected.

While it is accepted that the election result can be manipulated undetected if several components collaborate, we only address single manipulated voting system components. This is in general the case even for the JCJ/Civitas remote electronic voting scheme [11] which is one of the remote electronic voting schemes with the highest security guarantees. "Separation of duty" is implemented in voting schemes such that the responsible people behind these components (programmer, administrator and election officials) do not collaborate.

B. Results of the Analysis

In this subsection, we analyse whether POLYAS is affected by the threats identified above or whether it needs to be assumed that the corresponding threat does not exist, for example, by virtue of the election in question bearing a low public profile and therefore a low security risk. We present a summary of the results in Table II.

Table II
RESULTS OF THE ANALYSIS

Threat	Result of Analysis	Ok?
T1	Not met, treated as assumption according to [2]	(Ok)
T2	Detectable if applied in large scale (voters complain)	Ok
T3	Not met, but trust in VCI from BBS	No
T4	Detectable as ERS and VS control each other	Ok
T5	Not met, but trust in BBS	No
T6	Detectable if voter verifies certificates	Ok
T7	Not met, but trust in TC	No

Result concerning threat T1: POLYAS does not address the so-called trusted platform problem and correspondingly, there are no mechanisms implemented to meet viruses or malware on the voter's PC. Therefore, for elections using the current POLYAS software, it *needs to be assumed* that the PC is trustworthy. This assumption is acceptable given that POLYAS has been so far been used in elections facing a low security risk.

Result concerning threat T2: The attack that the VCI can be manipulated to send voting tokens to an attacker *is detected* by voters, assuming that they verify the SSL certificates carefully. When voters wish to cast votes and are falsely informed that they have already done so, they can file a complaint bringing this to the attention of the election officials. Note, it can be detected as vote-updating is not enabled in the POLYAS voting system.

Result concerning threat T3: In the current POLYAS implementation this attack is not addressed. Correspondingly, it is possible for a manipulated VCI to be modified in a way that a voter's selection for candidate *A*, for example, is sent as a vote for candidate *B* to the BBS. Therefore it *needs to be assumed* that the VCI received from the BBS is not manipulated.

Result concerning threat T4: This attack is addressed as the "separation of duty"-principle is implemented: ERS and VS control each other. In the protocol, the ERS receives a TAN from the voter. After verifying eligibility, it passes on these credentials to the VS for verification. Both servers must authenticate a voter positively before a voting token is generated. There are four different attack scenarios. First, the ERS only knows the hashed TAN values, therefore it cannot add new voters as corresponding TANs will not be known by the VS. Similarly, the ERS cannot vote on behalf of eligible voters as its database does not contain the corresponding TANs. In addition, prohibiting access to eligible voters by the ERS will be detected by voters (refer to result concerning threat *T2*). Prohibiting access to eligible voters by the VS will be detected by the ERS. In the fourth scenario, the VS could generate voting tokens and send them to the BBS to cast a vote, even if there was no voter request. However this would be detected by the ERS as it does not know the corresponding voting token but will be asked in *step (13)* (compare to Fig. 3) to delete it. Thus, all the different aspects of this threat *are detected* by the ERS (in case the VS is corrupt) and vice versa.

Result concerning threat T5: This attack is not addressed in the current POLYAS implementation. First, a manipulated BBS can encrypt and store a vote that differs from the content of the vote it received from a voter. Second, a manipulated BBS can change the content of the ballot box database by replacing encrypted votes with others. Note that the hash chains are available for verification but this is only applied if problems are detected. However, as a corresponding attack cannot be detected, the hash values would not be used for verification purposes. Correspondingly, it *needs to be assumed* that the BBS is trustworthy and stores what it receives without later modifying the database entries. Note that simply adding or removing votes in the database would be detected by comparing the number of voters in the electoral register.

Result concerning threat T6: POLYAS uses SSL to secure communication between entities. Under the assumption that voters verify the SSL certificate presented prior to proceeding with voter authentication and again before proceeding with vote casting, an attack violating the integrity of messages on the network *is detected* by the receiving component.

Result concerning threat T7: In the current POLYAS implementation this attack is not addressed. A manipulated TC can output an arbitrary election result as it is only verified that the number of votes in the ballot box is equal to the number of voters in the electoral register. Correspondingly, it *needs to be assumed* that the TC works correctly and is not manipulated. Note that the printout generated by the TC does not help reduce this assumption as it is produced by the TC itself.

Summary and Discussion of Result: This analysis shows that there is a need to place trust in single entities in the POLYAS remote electronic voting system, namely to trust the BBS and the TC. The corresponding risk of manipulated BBS and TC can be reduced by the Common Criteria evaluation and an accompanying compliance certificate according to the Protection Profile [2]. Should POLYAS, as one of the widely used remote electronic voting systems, be used for other types of elections and in particular for any parliamentary elections (e.g. as an additional channel to mail voting), this required trust in single entities would no longer be acceptable. However, this can be addressed by incorporating verifiability in such a way that voters and/or the public can verify that the corresponding threats (*T3, T5, T7*) were not executed. This verifiability would also be required by the court decision [5]. Besides the court decision, verifiability is an important requirement to enhance voter trust in election results, discussed in the research community since 1985 beginning with [12].

IV. VERIFIABILITY

In this section, we propose mechanisms to enable verifiability in the POLYAS remote electronic voting system. To do so, we first present the verifiability definitions used, according to [13]. We make a distinction between individual verifiability and universal verifiability.

Individual verifiability addresses the voter. The goal is to enable the voter to verify that

1) his vote is properly prepared and sent to the voting server, i.e. if the voter selects candidate *A* then candidate *A* is also sent and not candidate *B* [cast/sent as intended];
2) his vote is stored unaltered in the ballot box, i.e. if candidate *A* has been sent from the VCI then candidate *A* must be stored in the ballot box (in an encrypted manner) and not candidate *B* [stored as cast/sent]. Note, this covers two aspects, namely storing what has been received and no modification of the ballot box entries should be allowed later on.

Universal verifiability enables everyone to verify that all the votes stored in the ballot box are properly tallied. This includes verifying that decryption is done properly [tallied as stored]. Note that a remote electronic voting system that incorporates both types of verifiability provides end-to-end (E2E) voter verifiability. Kremer et al. [14] introduced eligibility verifiability, i.e. the public can verify that only eligible voters cast a vote and that they do so only once. This is not addressed in this paper as the corresponding threat *T4* is already secured in the current version by "separation of duty".

The following mapping shows that the above definitions cover exactly the issues identified in the security analysis. The first part of the individual verifiability definition allows the voter to verify whether an attack according to threat *T3* was executed. The threat *T5* is met if voters are able to verify that the vote has been stored as cast which is defined in the second part of the individual verifiability definition. Universal verifiability maps to threat *T7* as it enables the public to verify whether the votes in the ballot box are properly tallied.

A. Individual Verifiability Approach (cast as intended)

To achieve the cast as intended aspect of individual verifiability, the voter assesses the HTML source code of the election web page when the ballot is displayed. In doing so, he sees the data items passed to the system when clicking on each candidate (compare to Fig. 5, here '1' is sent for candidate 1). While this is generally possible it presents usability challenges. Not many voters know how to get access to the HTML code to check this information on web pages. In addition, if the ballot is very complex this process will be time consuming even if a voter has expert knowledge. Therefore, this verifiability should be done automatically and not manually. A corresponding tool is required and should be provided in future by independent trustworthy institutions such as universities. Note, even by providing this verifiability to the voter, POLYAS still remains receipt-free.

B. Universal and Individual Verifiability Approach (stored as cast)

We developed a verifiability tool to provide partial individual verifiability in terms of 'stored as cast' as well as universal verifiability in the POLYAS electronic voting system. This subsection first describes the tool and then provides an explanation which verifiability properties are now ensured and which corresponding threats are met.

```
1 <div class="candidate">
2       <div class="cand_img">
3       <p>Candidate 1</p>
4       <img src="images/candidate_1.jpg" width="73" height="97" alt="" />
5       </div>
6       <div class="cand_vote">
7       select this candidate 
8       <input name="votum" class="inp_check" type="checkbox" value="1"/></td>
9       </div>
10 </div>
```

Figure 5. HTML code relevant for verifiability

Description: The tool was developed using Java and integrates the Bouncy Castle Cryptographic APIs as well as some methods of the Apache Commons Codec API. The required inputs from the corresponding components or entities (for a screenshot of the test election see Fig. 6) are:

- Ballot box file from the BBS: This file contains names of the candidates and encrypted votes, block identifiers, block order, and status (refer to Table I).
- Hash value file from the ERS: These are the stored hash values.
- Keystore file from the TC: The keystore file contains the private RSA-key with which asymmetric encryption is realized. The keystore file itself and the private key are protected by passphrases.
- Passphrase of keystore file from election officials: The passphrase is required to access the keystore.
- Password of private key from election officials: The password has to be entered to get access to the private key.
- Maximal number of candidates from election officials: This number specifies the maximum number of candidates to cast a valid vote. This is necessary for the tool to distinguish a valid vote from an invalid one.
- Number of voters from electoral register: Number of online voters according to the electoral register.

Based on this data three different verifications are processed. First, the number of encrypted votes with status 'cast' in the ballot box is compared to the number of voters. Next, all hash values of the hash chain are generated by the tool based on the information in the ballot box file. Each value is compared with the corresponding one in the hash value file. In the third verification step the encrypted votes are decrypted vote by vote. To do so, the private key from the keystore is extracted and for each vote the AES key is first decrypted and then the encrypted vote is decrypted with this key. Based on the maximum number of candidates to cast a valid vote and the decrypted votes, the result is tallied and displayed. Now, this result needs to be manually compared with the result output by the TC. It is necessary to successfully verify all three steps.

Analysis: This tool provides universal verifiability as it decrypts and tallies the encrypted votes from the BBS. Thus, threat *T7* is met and trust in the TC to properly tally the result is no longer required. Regarding the second aspect 'stored as

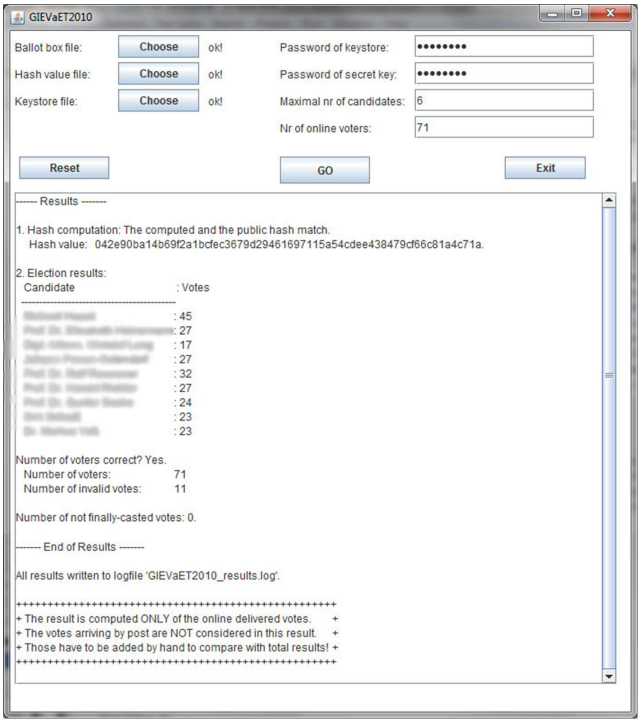

Figure 6. Verifiability tool interface (test election)

cast/sent' of individual verifiability only one aspect is covered by the tool. Under the assumption that ERS and BBS do not cooperate ("separation of duty"), stored encrypted votes cannot be manipulated undetected after having sent the corresponding hash value to the ERS. However, the other aspect is not covered, i.e. that it can be verified that what is stored is what is received. Correspondingly, only one aspect of threat *T5* is met. Thus, trust in the BBS is still partially required.

One might argue that the 'missing' verifiability aspect is already integrated in POLYAS as the BBS sends the stored selection in *step (11)* (refer to Fig. 3) back to the voter, and thus the voter can verify that the proper vote is stored in the ballot box. Unfortunately, in case of a malicious BBS this received vote does not allow any statements at the voter's side about which vote is stored in the database.

C. Discussion

Both proposed approaches provide an improvement for the POLYAS system as trust in the BBS is reduced and trust in the TC is no longer required (refer to Table III). Note, some of the previously defined threats are met, as not a single component can violate the integrity of the election result undetected, but as "separation of duty" is applied it needs to be assumed that corresponding two components do not cooperate, either ERS and VS or ERS and BBS. Therefore, it is important that the "separation of duty" is properly implemented, for example: different servers, different administrators, ideally different software companies. With the proposed approaches POLYAS does not become an E2E verifiable voting system since one step (stored as received) is not ensured. Further research is required to identify whether the system can be modified or extended in order to become end-to-end verifiable. However, as trust in components can be dramatically reduced with the proposed approaches, these should be applied in future. Moreover, partial verifiability seems to be a trend. Norway for instance also provides only partial verifiability due to privacy constraints [15]. Their system provides individual verifiability and universal verifiability is only accessible to observers and not the public.

Table III
RESULTS OF ANALYSIS OF EXTENDED VERSION

Threat	Result of Analysis	Ok?
T1	Not met, treated as assumption according to [2]	(Ok)
T2	Detectable if applied in large scale (voters complain)	Ok
T3	Detectable by HTML code verification	Ok
T4	Detectable as ERS and VS control each other	Ok
T5a	Not met, but trust in the BBS	No
T5b	Detectable by verifiability tool	Ok
T6	Detectable if voter verifies certificates	Ok
T7	Detectable by verifiability tool	Ok

V. USE IN GI ELECTIONS

In this section we describe the application of this verifiability tool in the GI 2010 election and the plans for future elections.

A. Election in 2010

Before the election in November and December 2010, we received the specification of the interfaces we required. During the development we made the following finding: For the encryption of votes, Micromata uses the Bouncy Castle Provider implementation of RSA and AES. Since the default mode and padding of this implementation is not compatible with the (standard) Sun provider or the FlexiProvider[3], up to now one can only use the Bouncy Castle Provider implementation for decryption of votes. If Micromata changes its use of the Bouncy Castle Provider to the mode and padding styles 'RSA/ECB/PKCS1Padding' and 'AES/CBC/NoPadding', then the Sun or FlexiProvider implementations could be used instead. This is desirable because of the independence of the implementation and the independence of the Bouncy Castle Provider. Further, we were allowed to observe the electronic tallying in Kassel and after the official procedure we received the necessary files and information to run the verifiability tool. During the tallying phase in Kassel and the discussion of the tool, the following findings were made: First, in the past the GI only published the number of votes per candidate for the elected candidates but not for the others in order to be polite to those candidates who only got few votes. This information cannot be kept secret if verifiability should be possible for everyone. Second, in the past only the result of the combined election results from both the electronic votes and the mail votes have been published but not in a distinguished manner. Again, this needs to be changed in order to enable verifiability of the electronic election result.

[3]Download is available at http://www.flexiprovider.de/.

B. Future GI Elections

For future GI elections, the verifiability tool needs to be extended and made available to anyone in order for them to also verify the election result. Furthermore, the interface for a verifiability tool will be made public so that any interested parties can implement their own tool and apply it. In order to enable everyone to run the election, the number of votes per candidate will be published by the GI for all voters as well as the result for the postal votes. Thus, the result of the electronic votes can be deduced. Furthermore, we plan to develop a tool for the first part of the individual verifiability to assist voters check the content of the html source code.

In addition, the GI agreed that the hash values are not only sent to the ERS but also published on the GI election web page. Thus, trust in the ERS is no longer required since the signed hashed values can continually be accessed from the GI web page by everyone. This was not realized for the 2010 election. Besides this, it has not yet been decided whether Micromata will change the encryption format in a way that other Java cryptography providers can be applied for the verifiability tool.

VI. Conclusion and Summary

Currently, there is a gap between remote electronic voting systems in use and those proposed in scientific papers. While the latter focus on different aspects of verifiability, most of the systems in use do not provide verifiability. By providing verifiability in the POLYAS remote electronic voting system we close this gap. To do so, we first described the POLYAS remote electronic voting system including applied procedures and protocols. We then analysed the system according to the integrity of election result requirements and identified those threats where one single entity needs to be trusted as this single component could manipulate the election result undetected. We have shown that these identified threats match the verifiability requirements 'cast as intended', 'stored as cast/sent' and 'tallied as stored'.

We proposed two different approaches to introduce verifiability in the POLYAS remote electronic voting system. The first enables the voter to verify that his vote is 'cast as intended'. As the manual verifiability is not very user-friendly, future work will be to implement a corresponding tool that does this automatically for the voter. The second approach partially covers the aspect 'stored as cast' since modifications after the computation of the hash values can be detected. It also covers the 'tallied as stored' aspect of universal verifiability. With the proposed verifiability approaches almost all threats are met and almost all aspects of verifiability are enabled. As future work, we will research a mechanism to enable voters to verify that their votes are properly stored in the hash value published at the ERS. In addition, we will address the eligibility verifiability requirement.

As a proof of concept this approach has been tested in the GI 2010 election. We have summarized our experiences in this paper, including the Java cryptography provider compliance and the publishing of so far unpublished information concerning the election result, that is, the partial result of mail votes and the number of votes per candidate who have not been elected. The GI decided to apply our proposed approach for future elections to enable verifiability for everyone. Therefore corresponding interface specifications will be published as well as the required information about the election result. In this way, future GI elections will make great advances towards verifiability and trustworthiness.

References

[1] K. Reinhard and W. Jung, "Compliance of polyas with the BSI Protection Profile - Basic Requirements for Remote Electronic Voting Systems," in *Proceedings of the 1st international conference on E-voting and identity*, ser. LNCS. Springer, 2007, pp. 62–75.

[2] M. Volkamer and R. Vogt, "Basic set of security requirements for online voting products," Common Criteria Protection Profile BSI-PP-0037, 2008. [Online]. Available: http://www.bsi.de/zertifiz/zert/reporte/pp0037b.pdf

[3] N. Menke and K. Reinhard, "Compliance of POLYAS with the Common Criteria Protection Profile - A 2010 outlook on Certified Remote Electronic Voting," in *Proceedings of the 4th International Conference on Electronic Voting 2010*, ser. LNI. Springer, 2010, pp. 109 – 118.

[4] J. Buchmann and A. Roßnagel, "Das Bundesverfassungsgericht und Telemedienwahlen," *Kommunikation und Recht (K&R)*, vol. 9, pp. 543–548, Sep 2009. [Online]. Available: http://pqkeylength.com/reports/reports/Buchmann_Rossnagel.Artikel.pdf

[5] Bundesverfassungsgricht (BVerfG), "Urteil des Zweiten Senats," *German Federal Court Decisions*, vol. 2 BvC 3/07 (1-163), Mar 2009. [Online]. Available: http://www.bverfg.de/entscheidungen/cs20090303_2bvc000307.html/

[6] M. Volkamer, *Evaluation of Electronic Voting: Requirements and Evaluation Procedures to Support Responsible Election Authorities*, ser. LNBP. Springer, 2009.

[7] Network Voting System Standards (NVSS), VoteHere Inc. Public Draft 2, 2002.

[8] V. Hartmann, N. Meissner, and D. Richter, "Online Voting Systems for Non-parliamentary Elections - Catalogue of Requirements," Physikalisch-Technische Bundesanstalt Braunscheig und Berlin, Laborbericht PTB-8.5-2004-1, 2004. [Online]. Available: http://ib.ptb.de/8/85/LB8_5_2004_1AnfKat.pdf

[9] Council of Europe, "Legal, Operational and Technical Standards for E-Voting. Recommendation Rec (2004)11 adopted by the Committee of Ministers of the Council of Europe and explanatory memorandum," 2004. [Online]. Available: http://www.coe.int/t/e/integrated_projects/democracy/02_activities/02_e%2Dvoting/01_recommendation/Rec(2004)11_Eng_Evoting_and_Expl_Memo.pdf

[10] Online-Wahlen Expertengruppe der Gesellschaft für Informatik, "GI-Anforderungen an Internetbasierte Vereinswahlen," GI, 2005. [Online]. Available: http://www.gi-ev.de/fileadmin/redaktion/Wahlen/GI-Anforderungen_Vereinswahlen.pdf

[11] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society - WPES '05*. ACM Press, 2005, pp. 61 – 70.

[12] J. D. Cohen and M. J. Fischer, "A robust and verifiable cryptographically secure election scheme," in *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1985, pp. 372–382.

[13] R. Gharadaghy and M. Volkamer, "Verifiability in electronic voting - explanations for non security experts," in *Electronic Voting 2010, EVOTE 2010, 4th International Conference*, ser. LNI, no. 167. Springer, 2010, pp. 151–162.

[14] S. Kremer, M. Ryan, and B. Smyth, "Election verifiability in electronic voting protocols," in *Proceedings of the 15th European conference on Research in computer security*, ser. LNCS. Springer, 2010, pp. 389–404.

[15] K. Gjøsteen, "Analysis of an internet voting protocol," Cryptology ePrint Archive, Report 2010/380, 2010, http://eprint.iacr.org/2010/380.