# Coercion-Resistant Internet Voting in Practice

Christian Feier, Stephan Neumann, Melanie Volkamer

Technische Universität Darmstadt / CASED
Hochschulstraße 10
64289 Darmstadt
{christian.feier, stephan.neumann, melanie.volkamer}@cased.de

**Abstract:** Internet voting continues to raise interest both among research and society. Throughout the last decades, many Internet voting schemes have been developed, each one providing particular properties such as receipt-freeness or end-to-end verifiability. One attractive scheme is the JCJ / Civitas scheme due to its property of making coercion attacks ineffective. Neumann and Volkamer [NV12] analyzed the scheme and identified significant usability issues. To overcome these drawbacks, the authors extended the original work by integrating smart cards. In a follow-up work, Neumann et al. [NFVK13] conducted a theoretical performance analysis for this extension and improved the extension towards its applicability in real-world elections. Their analysis left several real-world considerations open for future work. The present work addresses these gaps: We present a prototype implementation of the revised extension and assess its real-world performance. Based on this contribution, we are able to conclude that the revised extension is feasible to be used in real-world elections.

## 1 Introduction

Internet voting continues to attract interest both among research and society. Starting with Chaum's seminal work [Cha81], throughout the last decades, many Internet voting schemes have been developed. e.g. FOO [AF92], Helios [Adi08], and the Estonian Internet voting scheme [MM06]. One scheme of particular relevance is the JCJ / Civitas scheme [JCJ05, CCM08] due to its promised coercion-resistance property. Coercion is one of the crucial problems that accompanies the conduction of remote elections such as Internet voting [KV06]. While the JCJ / Civitas scheme addresses the highly valuable security requirements of coercion-resistance, the scheme lacks usability due to sophisticated credential handling as outlined by Neumann and Volkamer [NV12]. To overcome this shortcoming, Neumann and Volkamer revised the scheme. Their extension builds upon the integration of smart cards to simplify the credential handling to improve the usability. While their proposal certainly improves the JCJ / Civitas scheme from a usability perspective, it turns out that their extension does not take performance into account. As a consequence thereof, it includes several routines which cannot be implemented sufficiently performant on smart cards, e.g. the proof of well-formedness. In a follow-up work,

This work has been published in Lecture Notes in Informatics (LNI), Volume P-232, INFORMATIK 2014 Big Data Komplexität meistern: Erhard Plödereder, Lars Grunske, Eric Schneider and Dominik Ull, ISBN: 978-3-88579-626-8 

Neumann et al. [NFVK13] conducted an improvement of the extension towards its applicability in real-world elections by modifying or removing routines which do not run performant on smart cards. As a second part they made a theoretical performance analysis for the revised extension. Their work did not consider varying variables like network- and transmission time or smart card operations like copy data from non-persistent to persistent storage.

To address these shortcomings, the present work takes up the [NFVK13] scheme. As a contribution it analyzes the need of all changes made from [NFVK13] and provides some alternative ways to improve its performance. In addition to that, the present work provides ideas how to handle certificates and precises the calculations with respect to network traffic for 2048 bit encryption. As a last contribution a prototype including all made modifications will be presented.

The remainder of this work is structured as follows: In Section 2, we provide a short overview about the [NFVK13] scheme. In Section 3, we explain the made design decisions in terms of implementation details of the prototype followed by a short system design explanation in Section 4. In Section 5, we reanalyze the performance with respect to 2048 bit encryption blocks and the suggested certificate handling. Furthermore we make a practical test using our prototype and compare these results with the theoretical results. Later on in Section 6 we present the interfaces of our prototype and guide through an election. Section 7 concludes this work and possible directions for future research are outlined.

## 2 The NFVK13 Scheme Overview

The [NFVK13] scheme builds upon the extension by Neumann and Volkamer [NV12]. The scheme proofed the feasibility for real-world elections by optimizing the registration- and voting phase by changing and removing some unnecessary subroutines. In this section we summarize the [NFVK13] scheme.

**Setup Phase.** The supervisor initialises the election and publishes all details on the bulletin board. The *tabulation tellers* generate a distributed ElGamal election key $pk_{EK}$ and publish the corresponding public key on the bulletin board.

**Registration Phase.** As a first step a voter *v* personally consults the *supervised registration authority* (SRA). The authority checks that the voter is not under direct influence of any coercers. The voter is then requested to insert her smart card into the card reader and the SRA authenticates to the smart card as the SRA. In the next step she is asked to set her personal owner PIN. Afterwards the SRA stores the private supervised credential share $c^v_{SRA}$ on the voter's smart card which is a normal credential share as used by the registration tellers. The smart card will then calculate the coefficient $c_p = \frac{c^v_{SRA}}{PIN}$ and store it. This is the only value which is stored during this consultation. The voter leaves the SRA afterwards.
To a later moment in time the voter connects to the election website to finish the regis-

tration process. She is asked to choose her preferred registration tellers out of all available registration tellers. After this all preferred registration tellers, from now on called trusted registration tellers (TRT(v)), will be saved on the smart card. Afterwards a connection to each TRT will be established and the private credential share $c_{RT}^{v}$ along with an encryption $S_i' = \{c_{RT}^{v}\}_{pk_{EK}}$ and the used random $r$ will be directly forwarded to the smart card. At the same time the registration teller will publish the public credential share $S_i = \{c_{RT}^{v}\}_{pk_{EK}}$, which is a reencryption of $c_{RT}^{v}$, on the bulletin board. The smart card then verifies the DVRP proving that $S_i'$ is a reencrypion of $S_i$ from the bulletin board. After obtaining all private credential shares the smart cards calculates the credential factor

$$c_{factor} = c_p \cdot \prod_{i \in TRT(v)} c_{RT_i}$$

**Voting Phase.** After finishing the registration phase the voter can start the voting process. To do this she visits the election website. On this website she can select the candidates of her choice. After the selection is finalized the selection is forwarded to the smart card which then encrypts the selection using the election key $pk_{EK}$ and asks the voter to enter her owner PIN. The card then calculates the private credential as follows

$$c = PIN \cdot c_{factor}$$

If she entered the real PIN it will result in the voter's real credential to be calculated, if not a fake credential is generated. This credential will be encrypted using $pk_{EK}$ and calculates the proof of knowledge (PKCV) which shows that the voter knows both, the credential and the vote. Furthermore the smart card computes $h(\{vote\}_{pk_{EK}}||\{c\}_{pk_{EK}}||PKCV)$ as a verification code and displays this code on the card reader's display. The ballot will then be sent to all available ballot boxes.

**Tallying Phase.** In the tallying phase, all tabulation tellers retrieve the ballots from all ballot boxes and the public credentials stored on the bulletin board. Zero-knowledge proofs are verified, duplicates (due to vote-updating) and unauthorized votes (due to the use of fake credentials) are eliminated. Finally, encrypted credentials of remaining ballots are discarded and the respective encrypted votes are distributively decrypted. Each step of the tabulation tellers is publicly verifiable based on a set of zero-knowledge proofs.

## 3 Design Decisions

[NFVK13] made some recommendations we did not keep. In this section we explain the made design decisions.

**Multos vs. Java Card.** In [NFVK13] the calculations are on the basis of Java Cards and the timings from [BCGS09]. As proposed in [NFVK13] Java cards only offer some high level methods like RSA encryption, but do not offer any methods to directly calculate modular operations like addition, subtraction, multiplication and division on the crypto

co-processor. Due to this for example the modular multiplication needs to be calculated using a mapping on RSA CRT decryption. This means one modular multiplication on Java cards needs as long as one RSA CRT decryption which produces a big overhead. Multos instead offers direct crypto co-processor access using MEL assembler commands. Therefore on Multos cards it is possible to calculate modular operations directly on the crypto co-processor which results in better timings.
Due to the advantages over Java cards we switched for our implementation from Java- to Multos cards.

**Protocol Revision.** Neither [CCM08] nor [NV12] define protocols for the communication between registration teller and the bulletin board and communication between the smart card and the bulletin board. Furthermore at the beginning of the implementation the Multos C-API did not offer a mapping on AES. Therefore, the protocol has been changed to use RSA encryption and signatures to authenticate and exchange data with the registration teller. [CCM08] uses Needham-Schroeder-Loewe protocol (NSL) to authenticate to the registration teller and to exchange an AES key for further communication. The NSL protocol needs two RSA encryptions to authenticate and exchange the AES key. The proposed protocol change needs three RSA encryptions to authenticate and exchange all data. Assuming that one RSA encryption is at most as slow as the data communication using AES, the feasibility of the [NFVK13] protocol is shown if the feasibility with the modified protocol can be proven. In the protocol we use the following notation: The entities are the smart card ($S$), Registration Teller ($RT$) and BulletinBoard ($BB$). The used keys are the public ($pk_X$) and private ($sk_X$) RSA key of entity *X* and the public ElGamal election key ($pk_{EK}$). $\{T\}_K$ describes the encryption of *T* using the public key *K* and $Sig_K(T)$ is the signature of *T* signed with the private key *K*. Furthermore $N_X$ denotes the nonce generated by entity *X*, $ID_X$ is the ID of entity *X* and $s_i$ denotes the credential share of registration teller $i$ and $S_i$ is the ElGamal encrypted credential share using the public election key. A protocol step is denoted by X $\rightarrow$ Y: T which means that entity *X* sends the message *T* to entity *Y*. The modified protocol is described as follows:

1. S $\rightarrow$ BB: request registration teller certificate
2. BB $\rightarrow$ S: registration tellers certificate
3. S $\rightarrow$ RT: $N_V, ID_V$, trusted RTs, $Sig_{sk_V}(N_V||ID_V||$trusted RTs$)$
4. RT $\rightarrow$ BB: $\{N_{RT}, ID_V\}_{pk_{BB}}$
5. BB $\rightarrow$ RT: $pk_V, pk_{EK}, Sig_{sk_{BB}}(pk_V, pk_{EK}, N_{RT}, ID_V)$
6. RT $\rightarrow$ BB: $N'_{RT}, ID_{RT}, ID_V$, trusted RTs, $S_i, Sig_{sk_{RT}}(N'_{RT}||ID_{RT}||ID_V$trusted RTs$||S_i)$, $Sig_{sk_{RT}}($trusted RTs$, S_i, ID_{RT})$
7. RT $\rightarrow$ S: $\{(s_i, r)\}_{pk_V}, S_i, Sig_{sk_{RT}}(s_i||r||S_i)$
8. S $\rightarrow$ BB: $S_i$, $\{N'_V, ID_V,$ trusted RTs$\}_{pk_{BB}}$ $Sig_{sk_V}(S_i||\{N'_V, ID_V,$ trusted RTs$\}_{pk_{BB}})$
9. BB $\rightarrow$ S: $\{True/False, True/False, N'_V\}_{pk_V}$

In step 1 and 2, the specific registration teller certificate will be requested and loaded onto the card. In step 3 the card authenticates to the registration teller due to its knowledge of $sk_V$. In step 4 the registration teller requests the public key of $V$ which the bulletin board responds in step 5. The registration teller can verify the identity of the card. If the verification is successful it will send the trusted registration tellers and the encrypted credential share of voter $V$ to the bulletin board and sends the same encrypted credential share and the credential share to the voter in step 7. To save performance only the private credential share $s_i$ and used randomness $r$ are send encrypted. The public credential share is sent unencrypted. To ensure its integrity, the data is signed. Due to the fact the the resources of the smart card are limited, the signature of $s_i||r||S_i$ is generated so that the card does not need to store $\{(s_i, r)\}_{pk_V}$. After the card received the credential share and verified that the encrypted credential share is a reencryption it sends a request to the bulletin board in step 8. This request contains the trusted registration tellers and and the encrypted credential share. The bulletin board will respond with if the encrypted credential is on the bulletin board and if the trusted tion tellers are correct. The used nonce $N'_V$ which was sent encrypted in step 8 is used as an integrity check. This nonce $N'_V$ will be resent by the bulletin board in step 9. The smart card will check whether the nonce is the same and only accept the credential share of registration teller $i$ if the check passed successfully.

**PIN Handling.** To avoid side channel attacks to spy out the PIN code during the voting process [NFVK13] recommends to calculate coefficients $c_p$ and $c_{factor}$ so that the smart card does not need to know the real PIN but be able to calculate the credential depending on the entered PIN. As a drawback this way costs some performance. Through the security model in [NFVK13] the smart card needs to be trusted. Therefore it is not necessary that the card does not know the correct PIN as long as there is no way that the PIN leaves the card. Therefore the implemented prototype is saving the real PIN and compares it with the entered PIN. It will always compare the full PIN and will not stop at the first wrong digit. If the entered PIN is correct the PIN will be hashed using SHA-1 and the real credential $c$ will be encrypted. If the entered PIN is wrong the PIN will be hashed and instead of the real credential the hash value of the wrong PIN will be encrypted. Using this way the card will always perform the same steps independent of the entered PIN which will prevent side channel attacks. Additionally it is not necessary to calculate the coefficients $c_p$ and $c_{factor}$ which will result in a slightly better performance in the registration phase.

**Certificate Handling.** To ensure authenticity of messages from the authorities, namely the registration tellers and bulletin board, an authentic RSA key is necessary. A card should be used for several elections but due to the lack of space on the smart card only a small number of certificates could be stored. To solve this problem only one certificate called *Root Certificate* will be stored on the card. This Root Certificate could be set during the production process of the card and can not be changed after it is set once.
To initialise a secure communication to an authority the, authority's certificate needs to be loaded on the card at first. The card contains a slim DER parser[1] which extracts the necessary information from the certificate. To be authentic a certificate needs to be signed

[1] Distinguished Encoding Rules parser after RFC 5280 http://www.ietf.org/rfc/rfc5280.txt

from the *Root Certificate*, must be valid and must match a subjectDN pattern e.g. the common name CN must start with `RT` if the certificate belongs to a registration teller. Thereby, a trust chain will be built and only one certificate needs to be stored permanently on the card and there is no restriction on how many elections can be held with the card. As a drawback one should mention that the card needs to verify a certificate at every reload which may lower the performance.

## 4 System Architecture

In Figure 1, the system design is depicted. The responsibility of each component will be explained.

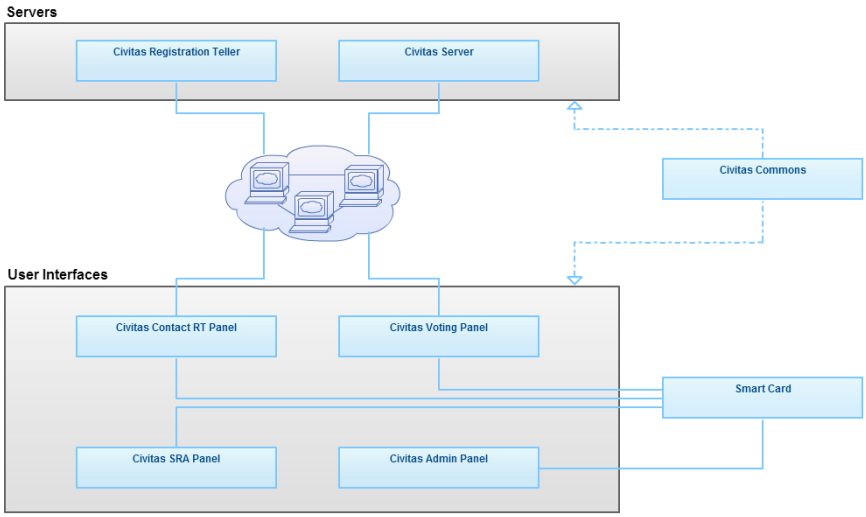


Figure 1: System Design

**Civitas Server:** The Civitas Server is a slim implementation of a real Civitas Server. It contains the bulletin board and offers an interface to communicate with the bulletin board. Furthermore this component includes the tallying module.
**Civitas Registration Teller:** The registration teller component is contacted by the voter to receive her private credential share. Several instances of registration teller components can run at different locations. The registration teller will be contacted remotely by the contact RT panel component.
**Civitas Contact RT Panel:** This component is a JApplet which offers a graphical user interface for the voter to contact a registration teller component and get her private credential share. This JApplet can directly communicate with the smart card.
**Civitas Voting Panel:** The voting panel is a JApplet which offers a graphical user interface for the voter to make her vote. It can directly communicate with the smart card.

**Civitas SRA Panel:** This component offers a graphical user interface for the supervised registration authority to authenticate to a smart card, allow the voter to set her owner PIN and set the supervised credential share to the smart card. Furthermore it transmits the current election parameters to the smart card. The component can directly communicate to the smart card.
**Civitas Admin Panel:** The administration panel offers a graphical user interface to initialize the smart card. It offers methods to set the UUID, set the root certificate and the signed card certificate. The admin panel is an optional panel and is not necessary to run an election. It can directly communicate with the card.
**Smart Card:** The smart card component is responsible to manage the credential shares, to generate the ballot which will be send to the bulletin board and to generate the verification code which will be shown on the card reader's display.
**Civitas Commons:** The commons package is used by all components. It includes commonly used classes like communication messages, offers an interface to communicate with the smart card and contains the whole cryptography methods.

# 5 Performance Analysis

In this section we concretize the formulas from [NFVK13] with respect to the changes made to the protocol described in section 3 and present results for a practical run from our prototype.

## 5.1 Smart Card Timings

The modular operations are *addition* ($a + b \bmod p$), *subtraction* ($a - b \bmod p$), *multiplication* ($a \cdot b \bmod p$), *modular exponentiation* ($a^b \bmod p$) and modular division $\frac{a}{b} \bmod p = a \cdot b^{p-2} \bmod p$, which corresponds to 1 subtraction, 1 multiplication, and 1 exponentiation. The measured timings for the operations are found in 1.

| Card / Operation | Addition | Subtraction | Multiplication | Expo. | Division |
|---|---|---|---|---|---|
| Java Card 1536 bit | 0.082 | 0.082 | 0.517 | 0.430 | 1.029 |
| Multos 1536 bit | 0.038 | 0.038 | 0.047 | 0.421 | 0.506 |
| Multos 2048 bit | 0.056 | 0.058 | 0.066 | 1.272 | 1.396 |

Table 1: Java and Multos Timing comparison

## 5.2 Concretize Formulas

**Registration Phase.** The formulas proposed in [NFVK13] only considered asymmetric cryptography operations like RSA and ElGamal encryption but did not consider the sym-

metric AES encryption which is used to secure the communication. The modified protocol described in section 3 only uses asymmetric encryption schemes to ensure a secure communication. Furthermore as stated in section 3, for a communication to an authority the certificate of the authority needs to be loaded on the card and then verified. To verify a certificate, a signature needs to be verified using `SHA1WithRSAEncryption`. The whole process to transmit a 2048 bit RSA key certificate with `SHA1WithRSAEncryption` signature, parse it and verify it takes approximately 4.4 seconds. For further reference, we define the whole certificate verification process as $t_{verify} = 4.403$ seconds.
We consider 2048 bit encryption which means that up to 255 bytes can be encrypted at once using RSA. Also we only analyse all steps in which the smart card is involved, namely step 3, 7, 8 and 9. To initiate the communication, the registration teller certificate needs to be loaded. To generate step 3 the card needs to generate one signature which results in one modular exponentiation and one certificate verification. Assuming that the credential share and the used randomness needs up to 255 byte, one RSA encryption and one signature generation is necessary to transmit the data. After decryption the reencrypted credential share needs to be validated by performing one ElGamal encryption. Summed up step 7 needs four modular exponentiations and one modular multiplication.
To perform step 8, communication to the bulletin board is required which means a new certificate needs to be loaded and verified. Furthermore, one RSA decryption and one signature verification is necessary which results in two modular exponentiations and one certificate verification for step 8. For step 9, one RSA decryption which means one modular exponentiation is required. After all registration tellers have been contacted, the credential $c$ can be generated by multiplying all credential shares $s_i$. In summary, the time needed throughout the registration phase is given as follows:

$$\begin{aligned} t_{registration} &= (1 \cdot t_{exp} + 1 \cdot t_{verify} + 4 \cdot t_{exp} + 1 \cdot t_{mul} + 2 \cdot t_{exp} + 1 \cdot t_{verify} + 1 \cdot t_{exp}) \\ &\quad \cdot |TRT| + t_{mul} \cdot |TRT| \\ &= (8 \cdot t_{exp} + 1 \cdot t_{mul} + 2 \cdot t_{verify}) \cdot |TRT| + t_{mul} \cdot |TRT| \end{aligned}$$

**Voting Phase.** In order to vote the credential and the vote needs to be encrypted and one proof of knowledge (PKCV) needs to be performed which results in four modular exponentiations and two multiplications for the two ElGamal encryptions and two modular exponentiations, two modular multiplications and two modular subtractions for the PKCV proof. Hence, the needed time is given as follows:

$$\begin{aligned} t_{voting} &= 4 \cdot t_{exp} + 2 \cdot t_{mul} + 2 \cdot t_{exp} + 2 \cdot t_{mul} + 2 \cdot t_{sub} \\ &= 6 \cdot t_{exp} + 4 \cdot t_{mul} + 2 \cdot t_{sub} \end{aligned}$$

### 5.3 Practical Results

The results from section 5.2 do not consider network traffic and transmission time between the client computer and the smart card. In this section we present our results from a live test of our prototype. The test system consists of four computers (one voter client, two registration tellers and one bulletin board) connected via a 100 Mbit LAN connection. The

used smart card is a Multos ML3 and the used card reader is an Omnikey 3821. The needed time will be measured after all user interaction is done, means for the registration phase after the trusted registration tellers are chosen and the 'proceed' button is pressed. For the voting phase it means after all candidates are chosen and the PIN is entered. To contact two registration tellers 46.782 seconds were needed, which means 23.391 per registration teller and the voting phase took 12.979 seconds. The theoretical solution took with respect to the timings from table 1 and $t_{verify} = 4.403$ results in a registration time of $38.228$ (i.e. $19.114$ seconds per registration teller) and a voting time of $7.809$ seconds. The theoretical analysis differs from the practical results. This is reasonable due to the fact that the theoretical analysis did not consider network and transmission traffic. Furthermore it did only consider modular operations like modular addition and the time needed to verify a certificate. Other processes on the card like hashing, copying data from non-persistent to persistent storage, reset data and many other operations could not be considered. We think that 30 seconds per registration teller and 13 seconds to vote are still feasible.

## 6 Interface

After the system has been implemented from a functionality perspective, user interfaces have been developed. These have been developed in cooperation with interface design experts and designers. In this section we present the first prototypes to guide a voter through the voting process. The interface is placed in a webbrowser using HTML as a markup language and a JApplet component to communicate with the card reader on the clients computer. Using this way the voter just needs to connect to a website and does not need to download additional software.

First, a voter connects to the election website (Figure 2). On this website the voter finds all election related information, namely the electoral register, the candidates, the authorities, the public authority keys, and the public election key. Furthermore, the voter can find all received ballots and encrypted credential shares on the bulletin board. There are several independent institutes which offer the voting software. The voter can decide over which institute she casts her vote.

On the welcome site (Figure 3) of the voting institute the voter is asked whether she already contacted her supervised registration authority. If not, she has to do this first and come back to this site. Furthermore she can choose if she already contacted her trusted registration tellers (for example in case of revoting). If she did not contact them before, she will be directed to the trusted registration tellers contact site (Figure 4). On this site a JApplet will be loaded. The voter will be instructed to insert her smart card into the card reader and choose her trusted registration tellers by simply clicking on them in the JApplet. After the voter made her choice, she confirms by the 'contact' button. Thereafter, the smart card contacts all chosen registration tellers and eventually returns a notifications once the registration process has been finished. The voter can start the voting process by clicking 'proceed'.

On the voting site (Figure 5) another JApplet will be loaded. The voter is requested to insert

her smart card into the card reader if not already done and to decide which candidate/s she wants to vote for. After pressing 'vote' she will be asked to enter the candidates ID into the card reader's PIN pad and submit it via the the card reader's submit button. The card asks for the owner's PIN and correspondingly calculates the ballot as described in section 2. After the calculation process is done, the card will send the ballot to the bulletin board. The corresponding verification code will now be displayed on the card reader's display.

After casting her vote, the voter is directed to the last site (Figure 6). This site informs the voter that her ballot is cast successfully. Furthermore it informs her that she can verify that her vote by visiting the bulletin board and check if her verification appears.

## 7 Conclusion and Future Work

Throughout the last decades, many Internet voting schemes have been proposed in the literature. One of these schemes is the coercion-resistant JCJ / Civitas [JCJ05, CCM08] scheme. As shown by Neumann and Volkamer [NV12], the scheme builds upon a number of trust assumptions and lacks in usability. Correspondingly, Neumann and Volkamer presented a proposal [NV12] to overcome these shortcomings by incorporating smart cards into the JCJ / Civitas scheme. Nevertheless, their work did not consider performance. In 2013, Neumann et al. [NFVK13] therefore analyzed the previously named proposal with regard to its feasibility. After slight protocol modifications, the authors were able to theoretically conclude the feasibility of the JCJ / Civitas extension.

As a final step of the previously outlined history, the present work proves the feasibility in a practical way. In the first part, we gave a short overview about the [NFVK13] scheme and afterwards outlined further design decisions made throughout the implementation of the scheme as well as the system design. In the second part, we refined the previous theoretical performance analysis from [NFVK13] and made a practical analysis. Third, we presented interface prototypes and provided a corresponding walktrough through the voting process. Summing up these contributions, we are able to conclude that that [NFVK13] is feasible to be applied in real-world elections.

In the future we plan to optimize the performance of the scheme. This can be done by exchanging space for performance e.g. by caching the bulletin board certificate. This would save one verification (currently 4 seconds) per registration teller in the registration phase. Furthermore optimizations for specific card types are imaginable e.g. by using the full non-persistent memory. This would save a few copy routines and would speed up the whole routine.

**Acknowledgment.** This work has been developed within the project ComVote, which is funded by the Center for Advanced Security Research Darmstadt (CASED), Germany. We also thank the reviewers for their valuable comments that helped to considerably improve the quality of this work.

## References


[Adi08] Ben Adida. Helios: Web-based Open-Audit Voting. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

[AF92] K Ohta A. Fujioka, T. Okamoto. A practical secret voting scheme for large scale elections. *J. Seberry and Y. Zheng, editors, Advances in Cryptology AUSCRYPT'92*, page 244251, 1992.

[BCGS09] Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard java card. In *ACM Conference on Computer and Communications Security*, CCS '09, pages 600–610. ACM, 2009.

[CCM08] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a Secure Voting System. In *IEEE Symposium on Security and Privacy*, pages 354–368. IEEE Computer Society, 2008.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *ACM Workshop on Privacy in the Electronic Society*, pages 61–70. ACM, 2005.

[KV06] Robert Krimmer and Melanie Volkamer. *Observing Threats to Voters Anonymity: Election Observation of Electronic Voting*. na, 2006.

[MM06] lle Madise and Tarvi Martens. E-voting in Estonia 2005. The first Practice of Country-wide binding Internet Voting in the World. In Robert Krimmer, editor, *Electronic Voting*, volume 86 of *LNI*, pages 15–26. GI, 2006.

[NFVK13] Stephan Neumann, Christian Feier, Melanie Volkamer, and Reto Koenig. Towards A Practical JCJ / Civitas Implementation. Cryptology ePrint Archive, Report 2013/464, 2013. `http://eprint.iacr.org/`.

[NV12] Stephan Neumann and Melanie Volkamer. Civitas and the Real World: Problems and Solutions from a Practical Point of View. In *International Conference on Availability, Reliability and Security (AReS 2012)*, pages 180–185. IEEE Computer Society, 2012.

# A Interface Mockups

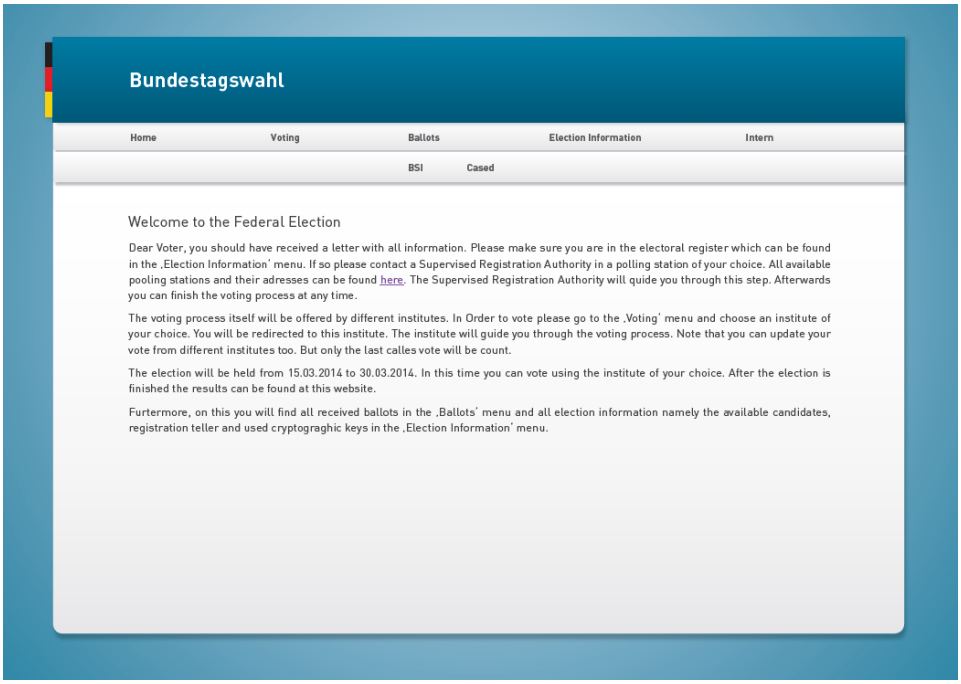


Figure 2: Bulletin Board

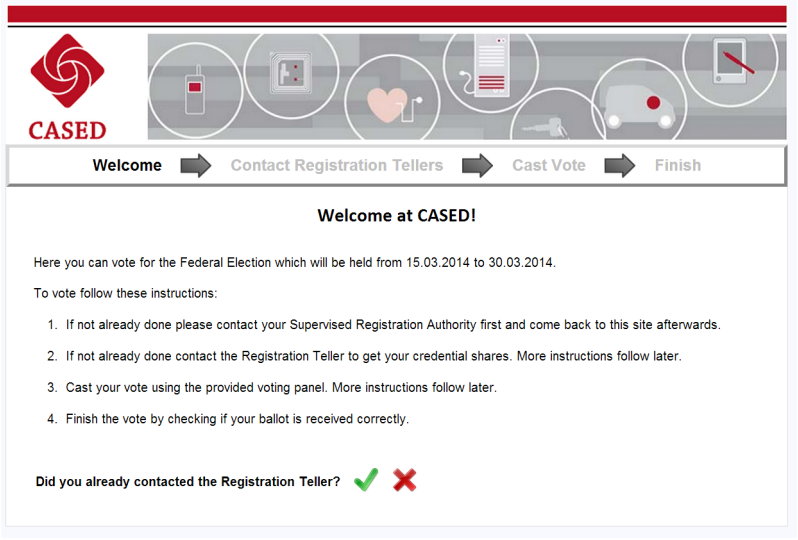


Figure 3: Vote Welcome Site

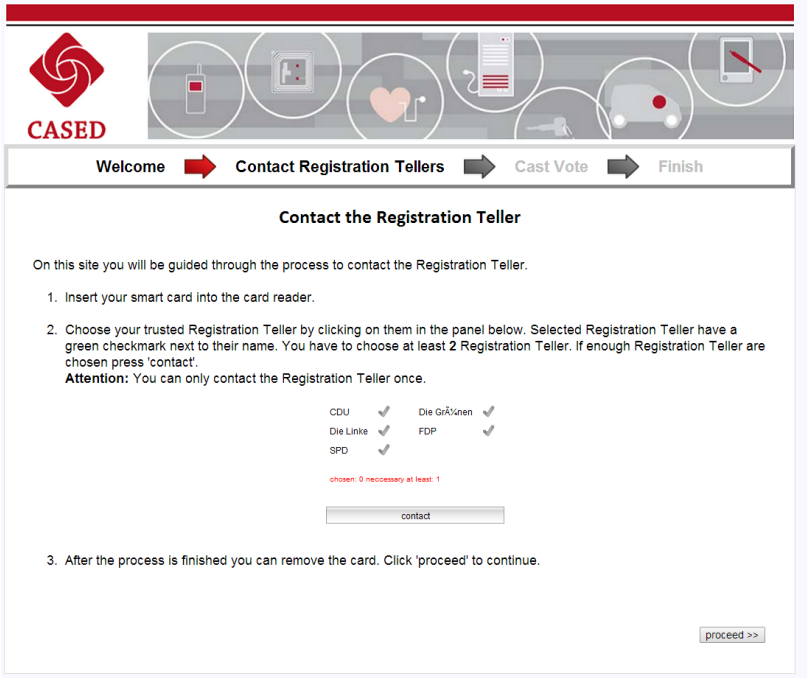


Figure 4: Contact Registration Teller Site

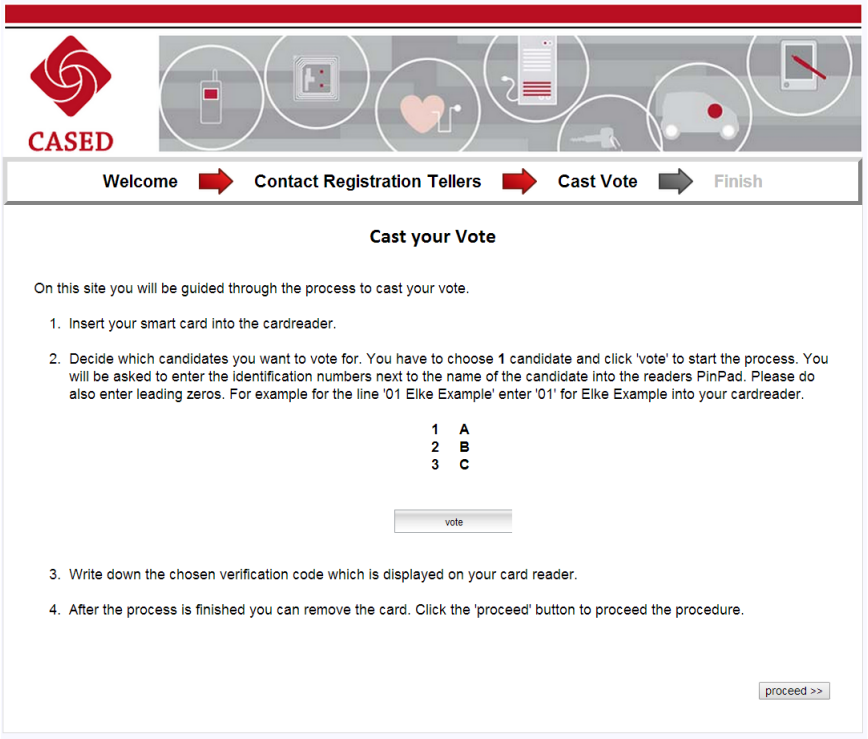


Figure 5: Cast Ballot Site

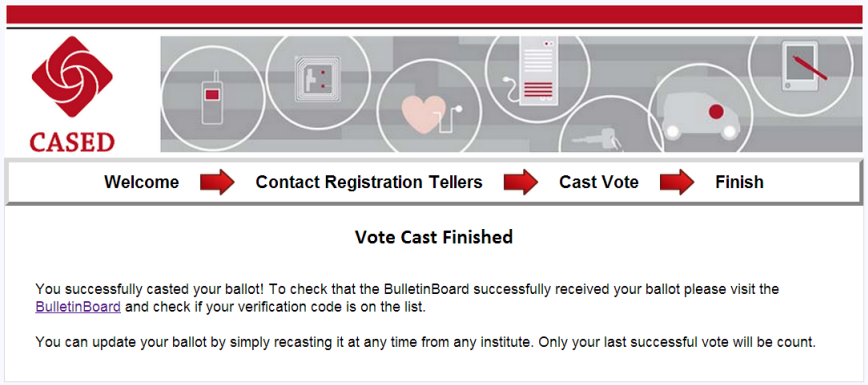


Figure 6: Vote Cast Finish Site