Civitas and the Real World: Problems and Solutions from a Practical Point of View

Stephan Neumann and Melanie Volkamer
CASED \ TU Darmstadt, Germany
Name.Surname@cased.de

***Abstract*—In the past, researchers have proposed many voting schemes that satisfy a wide range of security properties. These schemes often rely on strong trust assumptions and do not consider the voter sufficiently, which currently renders them inappropriate for usage in real-world elections. In this paper we focus on the voting scheme Civitas, which features provably strong security properties, such as end-to-end verifiability and coercion-resistance. We identify the strong trust assumptions and usability weaknesses of the scheme, which currently prevent its usage in real-world elections. Based on these results, we show how most of these strong trust assumptions can be implemented, e.g., by using eID cards in order to overcome Civitas' most critical usability problem, namely credential handling. Together with a voter-process description and a user-interface, we pave the way for the use of Civitas in real-world elections.**

I. Introduction

Over the last decades, many electronic voting schemes have been proposed. The majority of these schemes make use of sophisticated cryptographic concepts in order to provide a variety of security properties, among which the most important ones are end-to-end verifiability, ballot secrecy, receipt-freeness, and coercion-resistance. Unfortunately, ensuring all of these security properties comes at a cost and requires strong trust assumptions (TA) which cannot be addressed by means of organizational measures, e.g., a trusted voting environment, untappable channels, etc. Voters are therefore obligated to ensure the validity of such strong TAs. Furthermore, theoretical proposals often do not take voters into account and therefore impose an intolerable burden on them, for example, the ThreeBallot scheme [1] asks voters to also mark candidates for whom they do not vote. For an electronic voting scheme, to be considered for real-world elections, it is therefore crucially important to consider these practical aspects.

In this work, we show how to improve one of the most famous cryptographic voting schemes with respect to practical aspects, namely the Civitas voting scheme [2] presented by Clarkson et al. Civitas is an extension to the JCJ scheme [3], which provides end-to-end verifiability and coercion-resistance based on rigorous mathematical proofs. Civitas ensures coercion-resistance through distributed credential generation, which allows coerced voters to generate fake credentials in order to comply with the adversary's will, while only votes assigned to the real credential are tallied. In the past, Civitas has been investigated by many researchers from different points of view [4], [5] and several improvements have been proposed, e.g., in [6], [7]. However, strong TAs as well as the complex handling of real and fake credentials prevent its use in real-world elections.

In this paper, we first identify strong TAs, i.e., TAs that cannot be addressed by means of organizational measures and therefore need a concrete technical implementation. Furthermore, we discuss usability drawbacks of Civitas and present implementations of the determined strong TAs. In addition to other techniques, the concept of smart cards allows us to tailor our contribution towards usable credential handling within the Civitas scheme. In order to pave the way for a usable implementation of the Civitas scheme, we provide a voter-process description.

II. Civitas Voting Scheme

In 2005, the JCJ voting scheme [3] was the first voting scheme to satisfy coercion-resistance. The Civitas voting scheme was introduced in 2008 [2] and extends the JCJ scheme with respect to trust distribution and reduces its complexity. We base our approach on the version proposed in [7]. Due to the lack of space, we refrain from a description of the Civitas scheme, but rather point the reader to [7].

A. Entities

Each *voter* holds two key pairs, a so called registration key pair and designation key pair. The *supervisor* is in charge of naming registration, and tabulation tellers, fixing the ballot design as well as starting and stopping the election. The *registrar* administrates the electoral roll. *Registration tellers* independently generate private / public credential shares for all voters in the electoral roll. At the end of the election, *tabulation tellers* collectively decrypt and tally the cast votes. During the election phase, voters cast their ballots to the *ballot boxes*. Once the election is finished, all ballot boxes forward the stored votes to the *bulletin board*, where tabulation tellers access and process them.

B. Security Properties

Civitas ensures coercion-resistance, i.e., *voters cannot prove whether or how they voted, even if they can interact with the adversary while voting* and end-to-end verifiability as composition of universal verifiability, i.e., *any observer can verify that the votes stored in the ballot boxes are* *correctly tallied*, and individual verifiability, i.e., *each voter can verify that his vote has been cast as intended and stored as cast.*

C. Adversary Model and Trust Assumptions

Civitas assumes an adversary that can corrupt entities, thereby obtaining their secrets and / or controlling their behavior. The adversary furthermore controls the network, he may inject, read, and drop messages on public or anonymous channels. Civitas relies on the the following TAs:

TA 1. *The adversary is not capable of simulating the voter throughout the entire registration process.*

TA 2. *The adversary is only able to corrupt half of the registration tellers such that they do not treat the voter's credential properly and the adversary cannot see or use the channel from one trusted registration teller to the voter anyhow.*

TA 3. *The adversary cannot control or manipulate voters' voting environments containing the client-side voting software, the operating system as well as the hardware in use, e.g., by installing malware.*

TA 4. *The adversary cannot observe who sends messages to the ballot boxes.*

TA 5. *The adversary is not able to corrupt all ballot boxes and the adversary cannot drop the voters' votes towards the correct ballot box.*

TA 6. *The adversary is not able to corrupt more than k out of all n tabulation tellers.*

TA 7. *The adversary is restricted to probabilistic polynomial time computations and cryptographic primitives work.*

III. Analysis

In this section, we review Civitas' TAs from a practical point of view and determine those TAs that need to be implemented with special care before Civitas can be used in real-world elections. Afterwards, we study the protocol from the voter's perspective to identify usability problems.

A. Trust Assumptions Analysis

TAs 2 (first part), 5 (first part) and 6 are assumptions on election authorities. These assumptions can be implemented by organizational measures and scaled depending on the election form (e.g., each party could provide one registration teller, one ballot box and one tabulation teller). Hence, these assumptions are not critical with respect to Civitas' real-world application. TA 7 is a standard assumption for cryptographic systems and not critical for Civitas' real-world usage. As such, we do not address this assumption but rather rely on cryptographic concepts.

The remaining assumptions constitute a serious hurdle for the real-world application of Civitas as they require voters' specific actions or, at least partially, are under the voters' control and therefore need the voters' awareness about underlying concepts and threats. Hence, we intend to concretize these assumptions into clear concepts which lifts the burden from the voter's shoulders. We therefore show for each of the remaining assumptions why it is of central importance to implement them by means of procedural or technical measures.

Civitas assumes that the voter cannot be controlled by the adversary throughout the entire registration phase (TA 1). It is entirely under the voter's control to implement or violate this assumption. Hence, a voter could easily offer / could be forced by the adversary to stay with him throughout the registration phase.

Civitas' specification requires one untappable channel (TA 2). Untappable channels in general are a strong physical assumption. While this assumption is crucial to coercion-resistance, concrete implementations are missing.

Coercion-resistance can be easily violated if the voting environment is malicious, which is excluded by TA 3. Moreover, the stored-as-cast part of individual verifiability is obtained by assuming that the voting environment behaves properly. Hence, Civitas ensures that all cast votes are stored in the ballot boxes. Correspondingly, voters may never notice problems or even the absence of their vote in the final tally, which excludes voters' complaints by assumption. By assuming a fully trusted voting environment, Civitas furthermore ensures cast-as-intended verifiability. It is clear that individual verifiability and coercion-resistance strongly rely on the voter-side and the adversary could force the voter to use malicious hard- or software. We therefore aim to direct this assumption towards a more conceptual task less vulnerable to adversarial attacks.

Forced abstention attacks are explicitly excluded by anonymous channels (TA 4). An anonymous channel cannot be established by organizational measures as the sender should be explicitly hidden from the system, including server, observers, etc. Hence, it is the voter's responsibility to establish such a channel. Therefore, we intend to shift the responsibility away from the voter-side.

Finally, Civitas assumes that the voter casts his ballot to at least one correct ballot box (TA 5). While the existence of a correct ballot box can be addressed by organizational measures, casting a ballot to a correct ballot box needs to be ensured by the voter.

B. Usability Analysis

Civitas' coercion-resistance is achieved through a sophisticated processing of data: Once the voter acquired his private credential shares from the registration tellers, the voter is in charge of producing and managing both real and fake credentials: In the case where the voter is not under adversarial influence, the voter prepares a ballot by associating his real vote with the real credential. If the voter is coerced, the ballot is prepared with the adversarial vote associated with the fake credential. While from a theoretical point of view, the way coercion-resistance is ensured is innovative, maintaining coercion-resistance under these duties becomes a difficult task for the voter from a practical point of view.

IV. Concepts

The implementations of TAs, which we propose in the following sections, rely on the following concepts: We propose shifting Civitas' remote registration to a semi-supervised registration. The voter therefore first registers at a *supervised registration authority* and then completes his registration remotely. In addition, we will use the Benaloh challenge [8], a probabilistic game in which a prover (i.e. the voter-side voting software) can convince a voter that it encrypted the voter's vote correctly. The implementation of anonymous channels by anonymization networks has been studied intensively and many techniques exist. As one such technique, we propose to incorporate the TOR approach[1]. We also propose the use of eID cards that provide the following routines: The function `gen_keys` prompts the card to generate a registration key pair and a designation key pair. The public keys are returned to the issuing authority. The function `set_pin` prompts the cardholder to set a voting PIN. The functions `get_supervised_cred` and `get_remote_creds` are called in order to obtain and store private credential shares together with the corresponding DVR proofs on the card from the supervised registration authority and the remote registration tellers respectively. The function `prepare_ballot` prepares the voter's ballot based on his selection and his submitted PIN, i.e., if the voter submits his correct voting PIN, a valid ballot is prepared, otherwise the generated ballot contains a fake credential. The function `cast_ballot` casts the prepared ballot. As further feature of the card, it outputs the hash value of the ballot on the card reader's display. We propose that ballot boxes upon reception of ballots publish hash values of the received ballots on the bulletin board. In order to verify the presence of his vote, a voter has to search his ballot's hash value on the bulletin board. Note, it is of importance to use the browser search functionality rather than providing a server-side search feature. If the server would offer a search functionality, hash values could be arbitrarily added by the server.

V. System Description

This section is dedicated to a stepwise voter-process description based on the concepts of the previous section. The voter-process is enriched by a clear user-interface and lessons learned from the Helios voting interfaces discussed in [9]. An overview of the modified Civitas scheme is provided in Figure 1, the voter-process description in Figure 2, and the user-interface in Figure 3. In the voter-process description numbered arrows indicate the sequence of interactions throughout the protocol run. We omit the description of setup and tabulation phase as they remain unchanged from the original specification.

[1]https://www.torproject.org

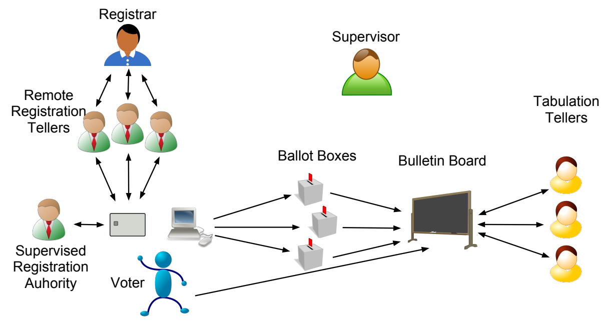


Figure 1. Sketch of the modified Civitas voting scheme.

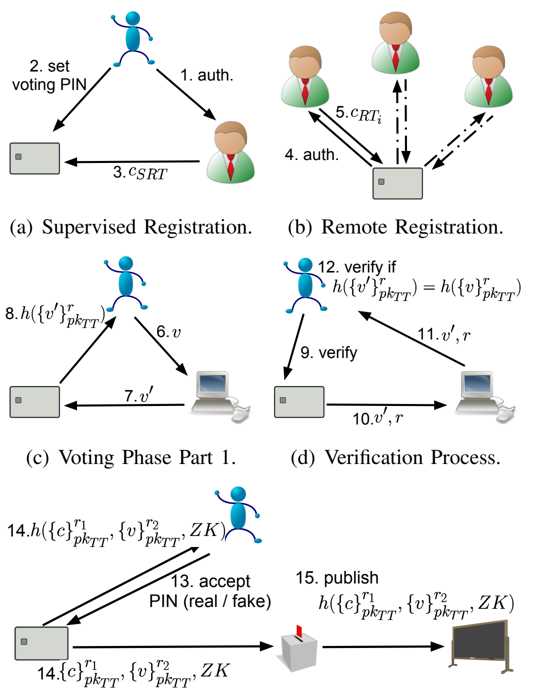
A. Registration

In order to register, the voter has to consult a supervised registration authority. It is important that this authority ensures that the voter is not controlled by the adversary during the stay, i.e., the adversary cannot see the voter's interaction with the election official and the card terminal. The voter then identifies and authenticates using his eID card. The authority then calls the function `set_pin` upon which the voter has to set his voting PIN used in the voting phase to release his real credential. The authority calls `get_supervised_cred` in order to store its private credential share c^V_{SRA} on the eID card (Figure 2(a)). Then, the voter leaves the supervised registration authority and can continue the registration process immediately or together with the voting phase. In the unsupervised part of the registration the voter inserts his eID card into his card reader and visits the election website, which loads a JavaScript. The voter obtains instructions and information about the voting process upon which he can proceed. Thereafter, the voter is asked to decide if he wants to finish the remote registration, proceed with the voting or verify the presence of a previously cast vote ((Figure 3(a))). If he decides to finish the registration phase, the JavaScript calls the function `get_remote_creds` and the registration proceeds according to the original Civitas specification (Figure 2(b)), hence, trusted registration tellers output their shares $c^V_{RT_i}$. The card computes the private credential as

$$c^V = c^V_{SRA} \cdot \prod_{i \in TRT(V)} c^V_{RT_i}.$$

B. Voting Phase

The voter then starts the voting process by selecting the candidate (or other options depending on the voting policy)


2. set voting PIN
1. auth.
3. c_{SRT}
5. c_{RT_i}
4. auth.
8. $h(\{v'\}^r_{pk_{TT}})$
6. v
7. v'
12. verify if $h(\{v'\}^r_{pk_{TT}}) = h(\{v\}^r_{pk_{TT}})$
11. v', r
9. verify
10. v', r
14. $h(\{c\}^{r_1}_{pk_{TT}}, \{v\}^{r_2}_{pk_{TT}}, ZK)$
13. accept PIN (real / fake)
15. publish $h(\{c\}^{r_1}_{pk_{TT}}, \{v\}^{r_2}_{pk_{TT}}, ZK)$
14. $\{c\}^{r_1}_{pk_{TT}}, \{v\}^{r_2}_{pk_{TT}}, ZK$

(a) Supervised Registration. (b) Remote Registration.

(c) Voting Phase Part 1. (d) Verification Process.

(e) Voting Phase Part 2.

Figure 2. Voter-Process.

encoded in v (Figure 3(b)) upon which the JavaScript calls the function `prepare_ballot`.

The card prepares a randomized encryption of the voter's choice obtained by the JavaScript and commits on that encryption by showing a hash value of the encryption on the reader's display (Figure 2(c)). The voter needs to write down this value for cast-as-intended verifiability. He is asked if he wants to verify the encryption process or if he is going to use the generated ballot (Figure 3(c)). In the case where he chooses to verify the encryption, the voter's choice, the used randomness and the vote are returned to the JavaScript as shown in Figure 2(d). The voter can charge an external institution (e.g. universities) with the verification process or verify a QR code (Figure 3(d)) with his smartphone. In both cases, the voter's choice and the randomness used to encrypt this choice are forwarded. The encryption process can be repeated independently and the voter can obtain evidence that the generated ballot contains his choice (Figure 3(e)). The voter thereafter has to restart the selection in order to maintain secrecy of his vote. If the voter confirms the process without verification, the card demands the PIN and prepares a ballot of the form $\langle \{c\}^{r_1}_{pk_{TT}}, \{v\}^{r_2}_{pk_{TT}}, ZK \rangle$, where depending on the voter's PIN, the real or a fake credential is used. The voter obtains the hash value of the entire ballot on the reader's display. He writes down that value and can use it for the purpose of stored-as-cast verifiability. Upon the `cast_ballot` call, the eID sends the ballot to all ballot boxes. The ballot boxes publish the hash value of the received data immediately on the bulletin board as shown in Figure 2(e) and Figure 3(f).

The concept we incorporate allows voters to verify immediately after vote-casting if their votes have been stored correctly. If not, they do not need to complain, but rather repeat the voting process from another voting environment.

VI. Discussion

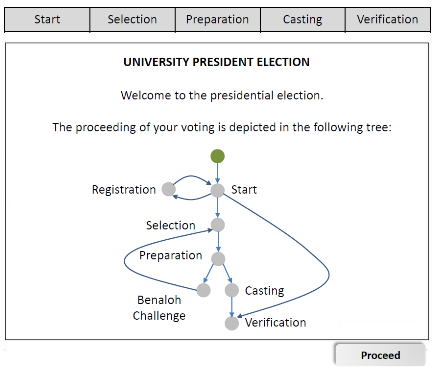
By conducting the registration partially supervised, TA 1 is realized, as a trusted registration authority can guarantee that the adversary cannot simulate an eligible voter in this registration step. Furthermore, the partial supervision allows to implement the untappable channel between the used eID card and one trusted registration authority as assumed in the second part of TA 2. Hence, TAs 1 and the second part of 2 can be replaced by New TA 1:

New TA 1. *Each voter trusts at least half of the remote registration tellers and the supervised registration authority with respect to credential distribution.*

In the voter-process description, voters still cast their votes over their machines, however the machine does not obtain information if a fake or the real credential is associated to that vote. Yet, our proposal can only be implemented if voters are equipped with smart card readers offering the possibility to enter PINs directly at the reader and to show hash values of 32 characters. Hence, with respect to coercion-resistance TA 3 can be replaced by New TA 2.

New TA 2. *Voters trust their eID cards and their smart card readers.*

When it comes to verifiability, a malicious voting environment may still alter or drop votes. The Benaloh challenge allows voters to verify that the cast ballot reflects their intention. The dependence on the voting environment is however reduced by additionally encoding the vote together with the randomness into a QR code. A smartphone can be used to read this code and compute the hash value internally. It would be enough if either the verification environment or the voter's smartphone is trusted. This would then lead to individual verifiability in the sense of cast-as-intended. In order to verify that the voter's ballot reached the ballot box as cast, the voter can verify that the hash value of the generated ballot appears in the list of hash values generated by the ballot boxes and correspondingly decide whether or not to re-vote from a more trustworthy machine. Relying on this concept also allows the voter to verify the correct handling of his vote in the non-verifiable anonymization network. It turns out that either the voting environment or the verification environment, i.e., the machine where the voter verifies the presence of the hash value of his cast ballot, needs to be trusted; hence we replace TA 3 by New TA 3 with respect to verifiability:

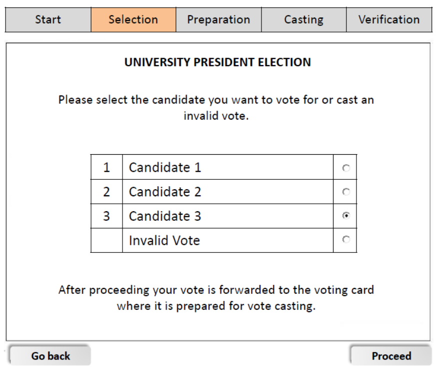


UNIVERSITY PRESIDENT ELECTION

Welcome to the presidential election.

The proceeding of your voting is depicted in the following tree:

(a) Welcome Page.



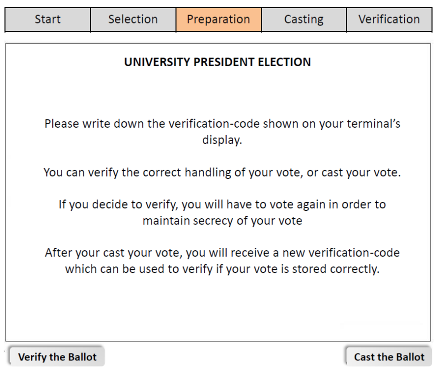
UNIVERSITY PRESIDENT ELECTION

Please select the candidate you want to vote for or cast an invalid vote.

1	Candidate 1	○
2	Candidate 2	○
3	Candidate 3	◉
	Invalid Vote	○

After proceeding your vote is forwarded to the voting card where it is prepared for vote casting.

(b) Candidate Selection



UNIVERSITY PRESIDENT ELECTION

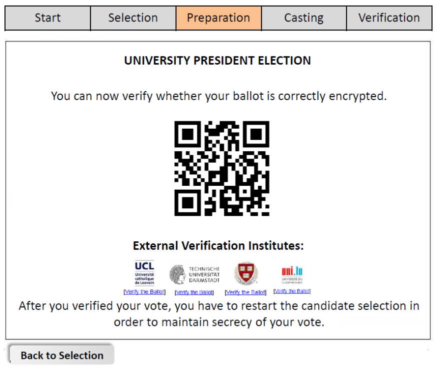
Please write down the verification-code shown on your terminal's display.

You can verify the correct handling of your vote, or cast your vote.

If you decide to verify, you will have to vote again in order to maintain secrecy of your vote

After your cast your vote, you will receive a new verification-code which can be used to verify if your vote is stored correctly.

(c) Verify or Cast your Ballot?



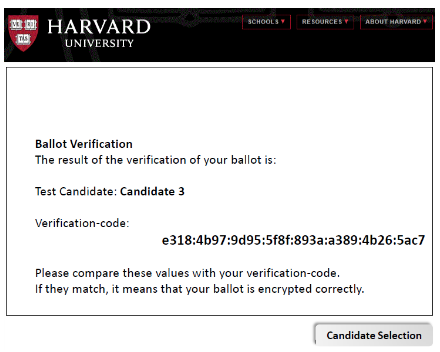
UNIVERSITY PRESIDENT ELECTION

You can now verify whether your ballot is correctly encrypted.

External Verification Institutes:

After you verified your vote, you have to restart the candidate selection in order to maintain secrecy of your vote.

(d) Benaloh Challenge.



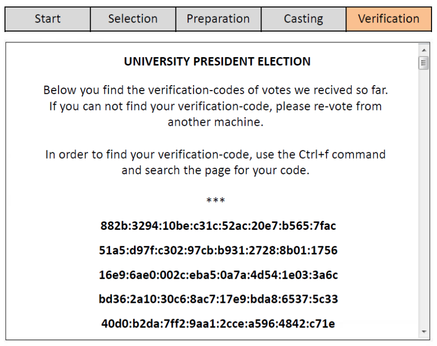
Ballot Verification
The result of the verification of your ballot is:

Test Candidate: **Candidate 3**

Verification-code:
e318:4b97:9d95:5f8f:893a:a389:4b26:5ac7

Please compare these values with your verification-code.
If they match, it means that your ballot is encrypted correctly.

(e) External Verification.



UNIVERSITY PRESIDENT ELECTION

Below you find the verification-codes of votes we recived so far.
If you can not find your verification-code, please re-vote from another machine.

In order to find your verification-code, use the Ctrl+f command and search the page for your code.

***

882b:3294:10be:c31c:52ac:20e7:b565:7fac

51a5:d97f:c302:97cb:b931:2728:8b01:1756

16e9:6ae0:002c:eba5:0a7a:4d54:1e03:3a6c

bd36:2a10:30c6:8ac7:17e9:bda8:6537:5c33

40d0:b2da:7ff2:9aa1:2cce:a596:4842:c71e

(f) Vote is stored correctly?

Figure 3. User-interface

New TA 3. *The adversary cannot control the voting environment and the verification environment at the same time.*

Publishing hash information after the vote casting allows voters to detect problems immediately and re-vote if their vote was lost. Hence, TA 4 is replaced by New TA 4.

New TA 4. *The adversary cannot control or manipulate all nodes in the anonymization network.*

Finally, the part of TA 5 which assumes that each vote is cast to one correct ballot box is ensured by sending the ballot to all ballot boxes, which leads to New TA 5:

New TA 5. *The adversary is not able to corrupt all ballot boxes.*

The absence of the voter's ballot's hash code would then raise the need to re-vote from another environment.

The presented concepts allow us to implement the critical TAs 1 to 5 identified in Section III, yielding five new TAs. While the original assumptions were the voter's responsibility, only New TA 3 is under the voter's control. All other new TAs can be ensured by organizational measures and are therefore not critical for Civitas' real-world application.

The presented implementation of TAs is tailored for usable credential handling; the use of smart cards fits this need exactly, as the computation capability of smart cards enables the card to come up with real or fake credentials automatically depending on the PIN submitted to the card without revealing any further details. To that end, voter-side assumptions are implemented by the presented concepts while furthermore in the particular the usage of eID cards and the voting PIN allow for a practical realization of credential handling.

VII. Related Work

JCJ and Civitas have been investigated thoroughly from theoretical points of view, e.g. [10], [4]. To the best of our knowledge, Civitas has been investigated from practical directions in only two works. In [6], Bursuc et al. proposed the Trivitas scheme. Their work mainly focuses on usability aspects and states that the complex mixing process of Civitas is not comprehensible to the average voter, furthermore trusted devices are needed to verify zero-knowledge proofs generated throughout the mixing. Therefore, they introduced trial credentials, which are indistinguishable from real credentials to all entities but the threshold set of decryption authorities. Their proposal addresses universal verifiability and moves that aspect towards individual verifiability without the need to verify complex zero-knowledge proofs about the correct behavior of election entities. However, credential handling is not addressed, which is crucial to maintaining strong properties as coercion-resistance. In fact, the distribution of trial credentials results in even more complex credential handling.

In [11], Mendes improved Civitas by promoting usability through the use of smart cards. Mendes removed the remote registration in favor of a fully supervised and trusted registration. While the author thereby addresses the problem of implementing strong trust assumptions, especially trust in the voting environment, his solution apparently bases the entire trust on the registrar. The distribution of smart cards remains unclear, and PIN codes to access smart cards are generated by another entity, which implicitly has to be trusted too. The author explicitly invented a further entity, the CodeCardReplier in order to increase stored-as-cast verifiability, which imposes another trust assumption on the scheme. It, however, turns out that cast-as-intended verifiability is not satisfied as the voting environment may still manipulate votes before handing them over to the smart card. Individual verifiability remains only partially satisfied, and the approach therefore fails to reduce the trust in the voting environment.

VIII. Conclusion and Future Work

The increasing interest in practical electronic voting schemes motivates our work in paving the way for Civitas' usage in real-world elections as a widely studied approach that relies on modern cryptographic concepts to achieve end-to-end verifiability and coercion-resistance. While Civitas has been at the center of many theoretical publications, the scheme unfortunately remains an abstract concept when it comes to real-world elections. First, Civitas relies on strong assumptions (TA), which render the scheme inappropriate to be used in real-world elections. Second, the complex handling of real and fake credentials assumes the voters' awareness of underlying concepts and threats. Third, even though Civitas' way of achieving coercion-resistance is indisputably innovative, the scheme lacks a voter-process that guides a voter, while at the same time maintaining the security properties.

In this paper, we first identified Civitas' TAs that cannot be addressed by organizational measures but rather are the responsibility of voters (namely five out of seven TAs). We presented an improvement of the Civitas scheme to implement TAs 1 to 5. While we introduced five new TAs, four of them can be addressed by organizational means. The remaining TA that is under the voter's control could be relaxed even further. Together with the presented voter-process description and the user-interface, this work lays the foundations for a real-world application of the Civitas scheme. We are aware of the fact that the requirements placed on the eID card are high demands. At this stage, however, we do not see a less resource-demanding implementation providing the same level of security. Even though the implementation presented herein relaxes most of the voter-side assumptions, not every problem is solved. An important task for the future is to provide stored-as-cast verifiability without relying on a trusted verification environment. One idea is to adapt the approach from [11] towards reliance on distributed authorities instead of a single CodeCardReplier. The acceptance of technology always comes with a clear user manual and process description. Therefore, remote voting schemes should not only consider the voter-process but also other election entities as lawyers, politicians and technical staff.

Acknowledgment

This paper has been developed within the project "ModIWa2" which is funded by the DFG.

References

[1] R. L. Rivest, "The ThreeBallot voting system," 2006, http://people.csail.mit.edu/rivest/Rivest-TheThreeBallotVotingSystem.pdf (last accessed March, 11 2012).

[2] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a secure voting system," in *IEEE Symposium on Security and Privacy*, May 2008, pp. 354–368.

[3] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *ACM Workshop on Privacy in the Electronic Society*, 2005, pp. 61–70.

[4] R. Kuesters and T. Truderung, "An epistemic approach to coercion-resistance for electronic voting protocols," in *30th IEEE Symposium on Security and Privacy*, 2009, pp. 251–266.

[5] B. Smyth, M. Ryan, S. Kremer, and M. Kourjieh, "Towards automatic analysis of election verifiability properties," in *Joint Conference on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security*, 2010, pp. 146–163.

[6] S. Bursuc, G. Grewal, and M. Ryan, "Trivitas: Voters directly verifying votes," in *Third International Conference on E-voting and Identity*, 2011.

[7] F. Shirazi, S. Neumann, I. Ciolacu, and M. Volkamer, "Robust electronic voting: Introducing robustness in civitas," in *International Workshop on Requirements Engineering for Electronic Voting Systems*, 2011, pp. 47 –55.

[8] J. Benaloh, "Simple verifiable elections," in *Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, 2006, pp. 5–5.

[9] F. Karayumak, M. Kauer, M. M. Olembo, T. Volk, and M. Volkamer, "User study of the improved Helios voting system interface," in *First Workshop on Socio-Technical Aspects in Security and Trust*, 2011, pp. 37–44.

[10] S. G. Weber, R. Araujo, and J. Buchmann, "On coercion-resistant electronic elections with linear work," in *Second International Conference on Availability, Reliability and Security*, 2007, pp. 908–916.

[11] J. M. B. da Silva Mendes, “Trusted Civitas: Client trust in Civitas electronic voting protocol,” Master’s thesis, 2011.