Experience from UNITA Elections: Reconciling Revote, E2E Verifiability and Low Coercion

Feng Hao[1], Luke Harrison[1], Saverio Veltri[2], Irene Pugliatti[2], Chris Sinclair[3], and Gareth Nixon[3]

[1] University of Warwick, UK
{feng.hao, L.Harrison.3}@warwick.ac.uk
[2] Eligo eVoting, Italy
{s.veltri, i.pugliatti}@eligovote.com
[3] Global Initiative Ltd, UK
{chris, gareth}@global-initiative.com

Abstract. This paper presents an experience of designing, building and deploying an online voting system for the Student Assembly elections in the UNITA Alliance with the following requirements. First, the system should allow voters to vote as many times as they wish before the election's closing time with only the last vote being counted (known as revote). Second, the system should allow end-to-end (E2E) verifiability. Third, the system should allow voters to cast votes under the minimum influence from external forces or coercion. Developing an online voting system to meet these requirements poses a unique challenge. In this paper, we present an online voting system for UNITA elections, based on a variant of the DRE-ip protocol to provide E2E verifiability with support for revote. The system adopts a two-server architecture and implements a separation of control between the two servers to protect the voter's anonymity. The first UNITA elections were successfully concluded in March 2025, providing a case study for reconciling revote, E2E verifiability and low coercion in a real-world setting. The use of verifiable online voting to empower students from different European universities to elect the Student Assembly also serves as a model for more inclusive democratic governance of a university alliance.

1 Introduction

UNITA (*Universitas Montium* in Latin) is an alliance of twelve European universities from seven countries (as of 2025, including France, Italy, Portugal, Spain, Romania, Switzerland and Ukraine)[4]. The UNITA universities comprise nearly 250,000 students and 21,000 staff. They are dispersed in rural, mountainous and cross-border regions across southern, western and central-eastern Europe.

The Student Assembly is a vital part of the UNITA alliance. It serves as a voice for the student body, ensuring that every issue, idea and feedback is heard and acted upon. The assembly comprises eight student representatives from each

[4] https://univ-unita.eu/

of the 12 universities. Among the eight, seven are elected through the electoral process, and one is nominated by the student council of each university.

The first elections for the representatives of the Student Assembly of UNITA took place from 24 to 26 March 2025. An online voting platform was developed to support the UNITA elections with the following requirements.

1. *Revote*: The system should allow voters to vote as many times as possible before the end time of the election, and only the last vote would be counted.
2. *E2E*: The system should allow end-to-end (E2E) verifiability, assuring ballots are 1) cast as intended; 2) recorded as cast and 3) tallied as recorded.
3. *Low-coercion*: The system should support a low-coercion model, in which voters would vote under the minimal influence from possible coercion.

While these requirements support certain security goals, they also introduce conflicts with others. For example, the support for revote helps mitigate the threat of coercion. But it also introduces a potential conflict with E2E verifiability, as the receipt for a confirmed ballot will be invalidated by revote. This adds complexity to the tallying process. How to reconcile the revote, E2E verifiability, and low coercion in a real-world election presents a unique challenge.

Another crucial consideration for the UNITA online voting system is *usability*. The system should be easy to use for ordinary voters without any technical background. This is perhaps the hardest challenge for deploying an E2E voting system since voters generally do not understand cryptography and are not familiar with the verification tasks defined in an E2E voting system.

This paper presents an experience of designing, building and deploying an online voting system for UNITA elections. First, we present a two-server architecture for the online voting system and adopt a separation of control between the two servers to protect the voter's anonymity. The voting system implements a variant of the DRE-ip protocol [22], with support for revote and weighted approval voting. This system was fully implemented and successfully used in the first UNITA elections in March 2025. We report our experience of deploying the system in a real-world setting, and discuss improvements and future work.

2 Architecture

The UNITA voting system adopts a two-server architecture (Fig. 1), consisting of a ballot casting (BC) server that interacts with voters through a web browser over HTTPS during the voting process and an E2E gateway in the back-end that provides end-to-end (E2E) verifiability as a service. The BC server and the gateway are independently managed by two different companies[5], ensuring a good separation of control. The BC server interacts with the gateway over a private and authenticated channel by invoking a set of application programming interface (API) functions provided by the gateway. The API is defined based on a variant of the DRE-ip protocol [22] with the added support for revote, as we will explain in detail in Section 3.

[5] Eligo eVoting and SEEV Technologies, respectively

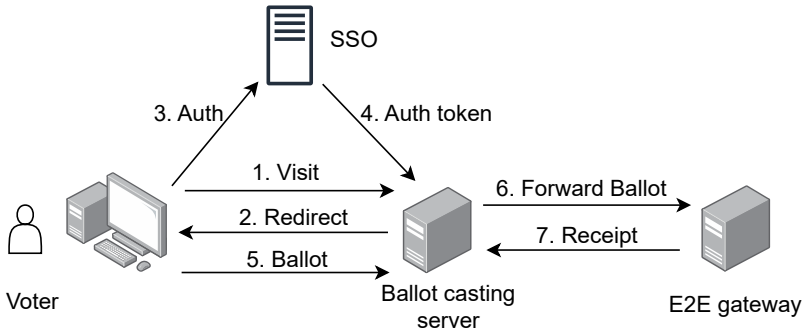

Fig. 1. A two-server architecture for UNITA online voting with single sign-on (SSO).

As shown in Fig. 1, the voting process consists of the following main steps.

1. The voter uses a web browser to visit an online voting address over HTTPS. Voters from each of the participating universities receive a unique address.
2. The BC server redirects the voter to their host institution for single sign-on (SSO) authentication.
3. The voter authenticates themselves to the SSO server over HTTPS.
4. The SSO server sends an authentication token to the BC server to indicate the authenticated status of the voter. Each voter is assigned a weight, which is an integer equal to or greater than 1. Normally, every voter has the same weight 1, but in some elections, they may have different weights.
5. After successful authentication, the voter accesses a list of candidates and chooses several within a range of the minimum and maximum selections.
6. The BC server forwards the ballot containing the voter's choice to the E2E gateway for verifiable tallying. The BC server does not store the ballot.
7. The gateway sends a receipt to the BC server following the E2E voting protocol (§3). The receipt contains a unique ballot ID, which the BC server stores with the voter's identity (this linkage is needed to process revote).

The gateway provides E2E verifiability as a service based on a variant of DRE-ip [22], an established E2E verifiable e-voting protocol. DRE-ip does not use any long-term decryption keys, hence avoiding the need for external tallying authorities (TAs) to manage those long-term keys. Instead, it only uses ephemeral keys for encryption and permanently deletes these keys as soon as the encryption is done. Only an aggregated value of the ephemeral keys is kept in the system to allow universal verification later. In the event of a system compromise, the tallying integrity is not affected due to E2E. The data leakage is limited to the minimum: only the partial tally at the time of compromise is revealed. Authors of DRE-ip name this TA-free system "self-enforcing e-voting" (SEEV). This "self-enforcing" feature particularly suits our two-server architectural design, as it allows a seamless integration between a vanilla online voting

platform and an E2E gateway with minimum changes to the former to build an E2E online voting system. Operations of the E2E gateway are automated.

However, we need to adapt the DRE-ip protocol in two aspects. First, the original DRE-ip protocol only supports plurality (first-past-the-post) without weighting, while UNITA elections use a weighted approval voting scheme permitting a minimum and a maximum number of selections. This extension can be done by using standard cryptographic techniques, as will be explained in Section 3. Second, the original DRE-ip protocol does not support revote, but UNITA elections require revote. This requires that rather than deleting the ballot and the ephemeral key immediately after the ballot encryption is done, we need to store them a bit longer during the voting phase in case the ballot will be overwritten later by revote. We explain more details of the protocol below.

3 The protocol

For the ease of description, we first explain a basic protocol (§3.1) for the single-candidate election, based on adding revote to DRE-i [22]. Later, we will explain extending this basic protocol to support weighted approval voting (§3.2).

3.1 Single-candidate election with revote

Let p and q be large primes that satisfy $q \mid p-1$. The protocol works in the subgroup $\mathbb{G}_q$ of order q of the group $\mathbb{Z}_p^\star$. Let g_1 be a generator of $\mathbb{G}_q$. In the UNITA elections, the protocol was implemented over the standard NIST P256 elliptic curve instead of a finite field for better efficiency, but the specification remains the same. The protocol consists of three phases: setup, voting and tallying.

Setup. To set up an election, the gateway receives election-specific information (e.g., title, start time, etc.) as input and derives a second generator g_2 through applying a one-way hash of g_1 together with election-specific data such that the logarithmic relationship between g_1 and g_2 is unknown to anyone. In addition, the gateway generates a random pair of digital signature keys based on ECDSA [23]. The private key will be used by the gateway for this specific election to prove data authenticity and integrity. The gateway returns g_2 and the public signature key to the BC server for publication on the bulletin board (BB), which is publicly readable, but writable only in an append-only manner [16].

Voting Phase. In this phase, the voter casts ballots through a web browser over HTTPS. The set of all ballots $\mathbb{B}$ consist of three types: the audited ballots $\mathbb{A}$, the cast ballots $\mathbb{C}$ and the revoted ballots $\mathbb{R}$, i.e. $\mathbb{B} = \mathbb{A} \cup \mathbb{C} \cup \mathbb{R}$. First of all, the voter needs to be authenticated through SSO. After the successful authentication, the voter casts a vote as follows.

1. The voter chooses a choice of 0 or 1 for the single-candidate election (corresponding to "No" and "Yes" respectively). The BC server receives the voter's choice and passes it to the gateway for tallying. The BC server does not store the voter's choice.

2. The gateway does not know the voter's identity. It assigns a unique ballot ID i for this vote $v_i \in \{0, 1\}$, generates random $r_i \in \mathbb{Z}_q^\star$, calculates

$$R_i = g_2^{r_i}, \quad Z_i = g_1^{r_i} g_1^{v_i}, \quad ZKP\{v_i\},$$

and provides a signed receipt including the unique ballot ID i and the ballot content R_i, Z_i, and $ZKP\{v_i\}$. $ZKP\{v_i\}$ represents a non-interactive zero-knowledge proof (ZKP), based on a 1-out-of-2 ZKP technique [11], to prove that given $\{R_i, Z_i\}$, v_i is either 0 or 1 without revealing which.
3. The voter is provided with the option to audit (cancel) or confirm the vote.

In case of audit:

4. The gateway adds i to $\mathbb{A}$, and returns a signed receipt of audit, clearly marked `audited`, including r_i and v_i.
5. The voter verifies that v_i in the provided receipt reflects her choice. If the verification succeeds, voting continues to Step 1; otherwise, the voter should raise a dispute immediately.

In case of confirmation:

4. The gateway adds i to $\mathbb{C}$, updates the tally and the sum:

$$t = \sum_{i \in \mathbb{C}} v_i \quad \text{and} \quad s = \sum_{i \in \mathbb{C}} r_i \ ,$$

and returns a signed receipt of confirmation, clearly marked `confirmed`. For elections without revote, the gateway deletes v_i and r_i immediately after the ballot confirmation. However, when the revote functionality is enabled, the gateway needs to keep v_i and r_i temporarily in case the voter chooses to revote later.
5. The voter finishes the voting session.

Re-voting. Assume the voter chooses to vote again. After the SSO authentication, the BC server determines that the voter has voted previously with a ballot ID i. It passes on the ballot ID i to the gateway to process revote.

1. The gateway removes i from $\mathbb{C}$, and adds it to $\mathbb{R}$. The gateway updates the tally and the sum:

$$t = t - v_i \quad s = s - r_i \ ,$$

and returns a signed receipt of re-voting, clearly marked `revoted` for the ballot i. The gateway permanently deletes r_i and v_i.
2. The voter follows the same steps as in the voting phase to cast another ballot under a new unique ballot ID $j \neq i$.

Tallying Phase. The election can be manually ended by an administrator or automatically ended by a pre-defined scheduler. Once the end of the election function is invoked, the gateway returns the final tally t and the final sum s. It permanently deletes all random values r_j and votes v_j, for $j \in \mathbb{C}$. After the election is finished, the gateway does not keep any sensitive private data about the election; the only data stored by the gateway is the public bulletin board *audit data*, which the BC server can retrieve as a whole if needed.

Candidate j	**Ciphertext**	**Well-formedness ZKP**
1	$(R_{i1}, Z_{i1}) = (g_1^{r_{i1}}, g_2^{r_{i1}} g_2^{v_{i1}})$	1-out-of-2 ZKP: $v_{i1} \in \{0, z\}$
2	$(R_{i2}, Z_{i2}) = (g_1^{r_{i2}}, g_2^{r_{i2}} g_2^{v_{i2}})$	1-out-of-2 ZKP: $v_{i2} \in \{0, z\}$
$\cdots$	$\cdots$	$\cdots$
n	$(R_{in}, Z_{in}) = (g_1^{r_{in}}, g_2^{r_{in}} g_2^{v_{in}})$	1-out-of-2 ZKP: $v_{in} \in \{0, z\}$
	$(X_i, Y_i) = (g_1^{\sum_j r_{ij}}, g_2^{\sum_j r_{ij}} g_2^{\sum_j v_{ij}})$	Range ZKP: $\sum_j v_{ij}$ in a range

Table 1. Encryption of ballot i in weighted approval voting. z denotes the weight.

Universal verification. The BC server has all the audit data, including the receipts, the final tally t and the final sum s, for publication on the bulletin board to support universal verifiability. Anyone with read access to the bulletin board can perform the following checks:

- verify that all the digital signatures are correct;
- verify that all the ZKPs are valid;
- verify that for all the audited ballots: R_i and Z_i included in the first part of the receipt are consistent with r_i and v_i included in the second part (and with the system parameters g_1 and g_2); and
- verify that the following equations hold:

$$\prod_{j \in \mathbb{C}} R_j \stackrel{?}{=} g_2^s \quad \text{and} \quad \prod_{j \in \mathbb{C}} Z_j \stackrel{?}{=} g_1^s g_1^t \ . \tag{1}$$

3.2 Extended Support for Weighted Approval Voting

The basic protocol can be extended to support multiple candidates in a weighted approval voting scheme. Assume there are n candidates. A voter is restricted to choose between $[n_{min}, n_{max}]$ candidates. The voting procedure remains the same as the basic protocol except that the ciphertext and the associated ZKP for each ballot are different. Table 1 summarizes the ballot encryption.

As shown in Table 1, the vote for each candidate j is encrypted in parallel using the basic "Yes/No" protocol. For each candidate, the vote is one of the two integers $\{0, z\}$ where z is the weight. In unweighted voting, each voter has the same fixed weight $z = 1$, but in weighted voting, voters may have different weights. The ciphertexts for all n candidates are aggregated below.

$$\begin{aligned}(X_i, Y_i) &= (\prod_j R_{ij}, \prod_j Z_{ij}) \\ &= (\prod_j g_1^{r_{ij}}, \prod_j g_2^{r_{ij}} g_2^{v_{ij}}) \\ &= (g_1^{\sum_j r_{ij}}, g_2^{\sum_j r_{ij}} g_2^{\sum_j v_{ij}})\end{aligned}$$

A range ZKP is used to prove that $\sum_j v_{ij}$ is one of the following values: $\{z \cdot n_{min}, z \cdot (n_{min} + 1), \ldots, z \cdot n_{max}\}$ without revealing which value. This can be realized by using a standard 1-out-of-n ZKP scheme [11].

4 UNITA elections

UNITA Student Assembly elections are held every two years, with elected representatives serving a two-year term. As a long-term project, it was decided that a phased approach should be taken to deploy E2E voting, allowing voters to gradually become familiar with the E2E concept and the verification tasks.

4.1 Phased deployment of E2E

E2E verifiability involves three verification tasks, namely, checking if a vote is 1) cast as intended, 2) recorded as cast, and 3) tallied as recorded. The first two are performed by each voter, and belong to *individual verifiability*, while the third one can be performed by anyone and hence belongs to *universal verifiability*.

These three verification tasks add different levels of burdens on the voter. The most difficult one is checking "cast as intended" [18], which is usually realized through voter-initiated auditing. In our system, the voter needs to check that the candidate's name in plaintext revealed on the cancelled ballot receipt matches their selection, and that the receipt is published on the bulletin board. This check is relatively straightforward but still requires a moderate effort. The verification on "recorded as cast" involves checking if the given receipt is published on the bulletin board. This is easier for voters, but the verification task still places a small, yet non-trivial burden. Finally, the verification on "tallied as recorded" places virtually no burden on ordinary voters, because they do not need to do it by themselves (anyone with computing skills can check it on behalf of all voters).

Based on the above consideration, a phased approach is adopted to deploy E2E voting in a gradual manner. The architectural design and the gateway implementation remain the same in all phases. The only difference lies in what verification tasks are required from the voter (see Table 2 for a summary).

- *Phase I*: In this phase, voters do not directly receive receipts. Instead, the BC server handles receipts on behalf of voters and publishes them on the bulletin board to allow universal verifiability on "tallied as recorded".
- *Phase II*: Voters receive receipts only for the confirmed ballots. This allows voters to check "recorded as cast", in additional to "tallied as recorded".
- *Phase III*: Voters receive all receipts from the gateway and perform the entire checks on "cast as intended", "recorded as cast" and "tallied as recorded".

Phase III will unleash the full potential of E2E voting, enabling every voter to be an auditor and to have the power to verify the tallying integrity of an election without trusting the online voting platform or any external authority. As such, it will provide the strongest guarantee about the tallying integrity. However, we may not reach there in one step. Voters are generally unfamiliar with E2E and the verification tasks that it requires. Phases I and II serve as transitional steps to provide voters with levels of verifiability in a progressive manner.

For the first UNITA elections held in March 2025, the Phase I approach was adopted for two reasons. First, in terms of usability, it is considered the easiest for voters (the time for educating voters about performing E2E verification

Verifiability	Type	Burden on voter	Availability
Cast as intended	Individual	Moderate	Phase III
Recorded as cast	Individual	Small	Phase II, III
Tallied as recorded	Universal	None	Phase I, II, III

Table 2. E2E Verifiability Available in a Phased Deployment.

before the first elections was limited). Second, it is related to the low-coercion consideration. In Phase I, all receipts are handled by the BC server (on behalf of voters) and published after the election to allow *universal verifiability*. Directly giving voters the receipts for *individual verifiability* would raise a concern about possible coercion. For example, with access to a voter's receipt of a confirmed ballot and the bulletin board, a coercer can check if that ballot has been overwritten by revote. The voter cannot deny revote in this case. Supporting deniable revote and E2E verifiability at the same time without adding hurdles to usability is a subject for further research.

4.2 The 2025 UNITA elections

The purpose of these elections was to choose student representatives for the UNITA Student Assembly. This assembly acts as a crucial link between the alliance's governance and students, providing feedback and proposals on various aspects of student interest, from decision-making processes to new project development. The following people from the UNITA Alliance are eligible to vote:

- students of all levels who, at the time of the elections, are regularly enrolled, either as current or out-of-course students, in the current academic year, including those enrolled in first-level study programs from the previous academic year who are awaiting graduation in the extraordinary session;
- incoming Erasmus students;
- PhD candidates; and
- students enrolled in specialisation courses.

An election was created for each of the participating universities with a unique voting address. Voters were uniquely identified by the BC Server using a JSON Web Token (JWT) provided by their host institution. The front end of the voting system, after the voter authentication, displayed the available lists and candidates (Fig. 2). Voters could choose up to one candidate from each list (Fig. 3). A blank ballot containing no selection of candidates was allowed. Once the vote selection was confirmed, voters could see whether they have voted or not and the timestamp of the latest vote cast. The voter could revote at any time before the end of the voting phase (Fig. 4).

The UNITA elections held between 26-28 March 2025 represented the first round of elections for the UNITA Alliance. Seven out of 12 universities participated. Out of a total of $190,290$ eligible voters, $15,870$ cast their votes, representing a 8.33% turnout (these results are provisional – the remaining universities in the alliance will organize their elections in the autumn of 2025).

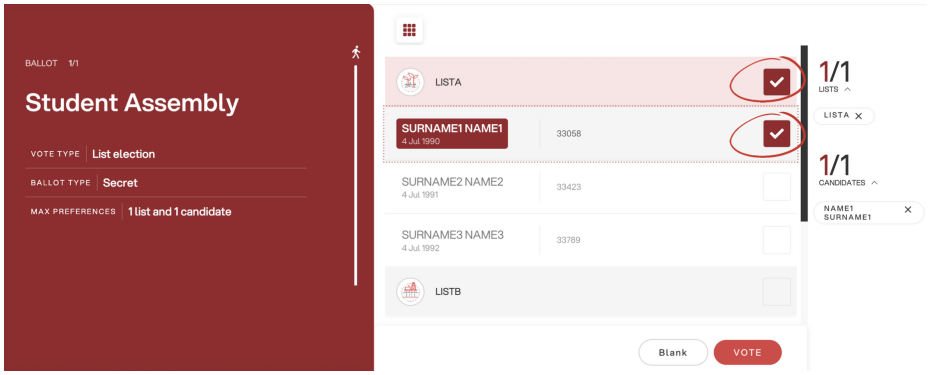

Fig. 2. Choosing candidates from a list.

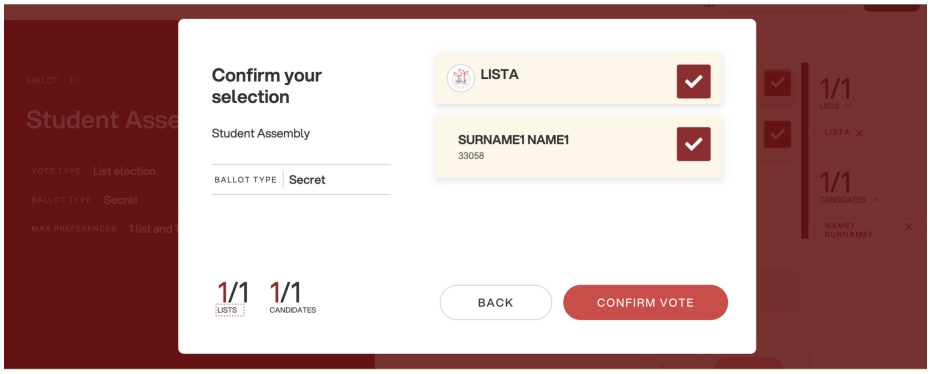

Fig. 3. Confirming selection of candidates.

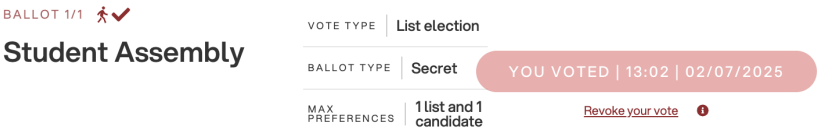

Fig. 4. Revoting after casting a confirmed ballot. The voter can revoke the ballot and cast another one at any time before the end of the election.

5 Discussions

The E2E verifiable e-voting protocol in Section 3 is based on adapting DRE-ip [22], an established E2E protocol that has been used in both local polling station voting [14] and remote Internet voting [12]. The extension of DRE-ip to support weighted approval voting uses standard cryptographic techniques. The setup, voting and tallying procedures are the same as in the original DRE-ip.

The gateway adopts the same self-enforcing tallying method as in DRE-ip to encrypt ballots by using only ephemeral keys. In order to support revote, the gateway cannot delete the ephemeral key immediately after the encryption of each ballot in case the voter decides to revote later. This requires protection of the ephemeral secrets for a temporary period of time, which can be done by using a hardware-based encryption method [4]. When the election is finished, any sensitive private data about the election will have been permanently deleted; the only data remaining at the gateway will be the public bulletin board data. The gateway has no access to the voter's identity at all, whilst the BC server stores only the linkage between the ballot ID and the voter's identity. The separation of the BC server and the gateway serves to protect the voter's anonymity.

Further measures to strengthen anonymity are possible, e.g., by removing the ballot ID and the voter identity linkage stored at the BC server. They involve additional secrets with different trade-offs. We briefly highlight two below.

1. *Revote passcode.* Let u be the user's identity. We may introduce a revote passcode p, which can be randomly generated by the BC server and emailed to the voter after they have cast a confirmed ballot with the ballot ID i. The BC server does not store p or u; instead, it stores a tuple $\{H(u,p), i\}$ where H is a one-way hash function. When the user u logs on to revote, they need to provide the revote passcode p, based on which the BC server can compute $H(u,p)$, retrieve the ballot ID i, and process revote accordingly. Compromising the data stored on the BC server does not reveal the anonymity since the attacker does not know p (which is kept by the voter). As a trade-off, this solution adds a burden for the voter to keep and supply p when needed.
2. *Hardware module.* Let k be an HMAC key [23]. We may use a hardware security module (HSM) in the cloud to generate k, which never lives outside the module. The BC server stores the tuple $\{\text{HMAC}_k(u), i\}$ in the database. When the voter u logs on to revote, the BC server queries the HSM to obtain $\text{HMAC}_k(u)$, and then retrieves the ballot ID i from the database look-up. A compromise of the server database does not reveal any linkage between i and u, without knowing the secret key k. This solution does not add any burden to the voter, but adds a significant cost for using HSM.

6 Related work

Various online voting systems have been designed in order to reconcile the support for revoting, verifiability, and minimal coercion. Helios [2] is a prominent

online voting system[6] which initially utilized the Sako-Kilian mix-net [21] in order to tally ballots. The later 2.0 version changed to use homomorphic tallying [3]. Helios is E2E verifiable, and adopts Benaloh's voter-initiated auditing technique [5] to assure that ballots are cast as intended. In 2009, the Université catholique de Louvain (UCL) trialed Helios 2.0 as part of an election to determine a new University President [3]. The UCL trial featured revoting: voters were permitted to cast as many ballots as they desired, but only the final cast ballot would be displayed on the bulletin board and added to the tally. Attacks including vote buying or voter bribery were considered to be out of scope [3]. Unlike our system, Helios requires a set of tallying authorities to manage shares of a long-term decryption key. As reported by the Helios authors, finding such authorities proved to be "particularly difficult".

Civitas [7] is an online voting system designed for a variety of election types including those where voters can choose a single candidate (plurality voting), a range of candidates (approval voting), or a preference list of candidates (ranked voting). Civitas assumes a strong adversary that can demand a voter to take a particular action such as submitting a specific ballot or nothing at all. As such, Civitas aims for *coercion resistance* and looks to achieve this by refining the Juels, Catalano, and Jakobsson (JCJ) [15] voting scheme. This scheme allows voters to be given a fake credential, which, when used during vote casting, results in a ballot that is never added to the tally. The voter receives a real credential during registration and the coercer is unable to distinguish a ballot cast using either credential. Civitas also permits a voter to revote multiple times per credential, however the JCJ scheme does not provide coercion resistance in this setting where a voter may be instructed to cast further ballots after the initial ballot under coercion [17]. Furthermore, the mixed use of real and fake credentials poses usability challenges to voters [19].

Belenios [9] is an online voting system[7] built upon Helios. The main difference between both systems is that Belenios aims for *eligibility verifiability*. In Belenios, anyone can check that the ballots were cast by legitimate voters, whereas in Helios, it is possible for a dishonest bulletin board to append ballots without being noticed [9]. Belenios also provides assurance that ballots are recorded as cast and tallied as recorded. Following the initial description of Belenios [9], several revisions have since been published. A receipt-free extension to Belenios, called BeleniosRF, was proposed in 2016 [6]. Later, in 2022, Belenios was extended with additional features including support for weighted approval voting and ranked voting [10]. In 2023, Belenios-CaI was proposed to feature cast as intended verifiability [8]. Belenios allows a voter to recast their ballot as many times as desired, with only the final ballot counting towards the tally. This requires the voting server to store a link between each voter and their corresponding key for verifying signed ballots; if this link is not stored, then a corrupt voter and credential authority can perform a ballot stuffing attack [9].

[6] https://vote.heliosvoting.org/
[7] https://www.belenios.org/index.html

Following these voting systems, several works have since attempted to pivot from the low-coercion model and instead detail systems that provide stronger guarantees of coercion resistance for online elections. Many of these works follow the model of *deniable revoting*, where a coercer cannot tell how many times their target chose to recast their ballot or that the ballot was even recast at all. Deniable revoting maintains the convenience of revoting, should a voter decide to change their vote of their own accord, whilst ensuring that any attempts at coercion by exploiting the revoting mechanism are not successful. Systems which aim to provide deniable revoting include VoteAgain [17], DeVoS [20], and Loki [13]. These systems achieve deniable revoting by providing a cover of dummy or noise ballots on the bulletin board that makes any revoted ballot indistinguishable from any other ballot. Additionally, Achenbach et al. [1] demonstrate that the JCJ scheme can be extended to feature deniable revoting. We note however that none of these proposals maintain E2E verifiability alongside deniable revoting. For future work, it would be beneficial to explore whether deniable revoting can be supported without compromises to E2E verifiability and usability.

7 Conclusion

We proposed a variant of the DRE-ip protocol with support for revote and weighted approval voting. Furthermore, we proposed a two-server architecture and a separation of control between the two servers to protect the voter's anonymity. Based on the proposed protocol and the two-server architecture, we built an E2E verifiable online voting system for the UNITA Student Assembly elections. The deployment of the system follows a gradual three-phase approach, with the first phase supporting only universal verifiability, the second phase adding partial individual verifiability and the third phase providing the full E2E verifiability. Finally, how to achieve deniable revote and E2E verifiability without adding hurdles to usability was proposed as an open topic for future research.

Acknowledgments

We thank anonymous reviewers for their helpful comments. Feng Hao and Luke Harrison are supported by the Engineering and Physical Sciences Research Council grant (EP/T014784/1).

References

1. D. Achenbach, C. Kempka, B. Löwe, and J. M"uller-Quade. Improved Coercion-resistant Electronic Elections through Deniable Re-voting. {*USENIX*} *Journal of Election Technology and Systems ({JETS})*, 3:26–45, 2015.
2. B. Adida. Helios: Web-based open-audit voting. In *USENIX Security Symp.*, volume 17, pages 335–348, 2008.

3. B. Adida, O. de Marneffe, O. Pereira, and J.-J. Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of Helios. In *EVT/WOTE'09*, page 10. USENIX, 2009.
4. R. J. Anderson. *Security engineering: a guide to building dependable distributed systems*. John Wiley & Sons, 2010.
5. J. Benaloh. Simple Verifiable Elections. *EVT*, 6:5–5, 2006.
6. P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo. BeleniosRF: A Non-interactive Receipt-free Electronic Voting Scheme. In *ACM CCS*, pages 1614–1625, 2016.
7. M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a Secure Voting System. In *IEEE Security & Privacy*, pages 354–368. IEEE, 2008.
8. V. Cortier, A. Debant, P. Gaudry, and S. Glondu. Belenios with Cast as Intended. In *Financial Cryptography*, pages 3–18. Springer, 2023.
9. V. Cortier, P. Gaudry, and S. Glondu. Belenios: A Simple Private and Verifiable Electronic Voting System. *Foundations of Security, Protocols, and Equational Reasoning: Essays Dedicated to Catherine A. Meadows*, pages 214–238, 2019.
10. V. Cortier, P. Gaudry, and S. Glondu. Features and Usage of Belenios in 2022. In *E-Vote-ID*. University of Tartu Press, 2022.
11. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *Crypto'94*, volume 839 of *LNCS*, pages 174–187. Springer, 1994.
12. H. Druliac, M. Bardsley, C. Riches, C. Dunn, L. Harrison, B. Roy, and F. Hao. On the feasibility of e2e verifiable online voting–a case study from durga puja trial. *Journal of Information Security and Applications*, 81:103719, 2024.
13. R. Giustolisi, M. S. Garjan, and C. Schuermann. Thwarting Last-Minute Voter Coercion. In *IEEE Security & Privacy*, pages 3423–3439. IEEE, 2024.
14. F. Hao, S. Wang, S. Bag, R. Procter, S. F. Shahandashti, M. Mehrnezhad, E. Toreini, R. Metere, and L. Y. Liu. End-to-end verifiable e-voting trial for polling station voting. *IEEE Security & Privacy*, 18(6):6–13, 2020.
15. A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Privacy in Electronic Society*, WPES'05, pages 61–70. ACM, 2005.
16. A. Kiayias, A. Kuldmaa, H. Lipmaa, J. Siim, and T. Zacharias. On the security properties of e-voting bulletin boards. In *International Conference on Security and Cryptography for Networks*, pages 505–523. Springer, 2018.
17. W. Lueks, I. Querejeta-Azurmendi, and C. Troncoso. {VoteAgain}: A Scalable Coercion-resistant Voting System. In *USENIX Security*, pages 1553–1570, 2020.
18. K. Marky, O. Kulyk, K. Renaud, and M. Volkamer. What did I really vote for? On the usability of verifiable e-voting schemes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.
19. L.-H. Merino, A. Azhir, H. Zhang, S. Colombo, B. Tellenbach, V. Estrada-Galiñanes, and B. Ford. E-Vote Your Conscience: Perceptions of Coercion and Vote Buying, and the Usability of Fake Credentials in Online Voting. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3478–3496. IEEE, 2024.
20. J. Müller, B. Pejó, and I. Pryvalov. Devos: Deniable Yet Verifiable Vote Updating. *Proceedings on Privacy Enhancing Technologies*, 2024.
21. K. Sako and J. Kilian. Receipt-free mix-type voting scheme. In *EuroCrypt'95*, volume 921 of *LNCS*, pages 393–403. Springer, 1995.
22. S. F. Shahandashti and F. Hao. DRE-ip: A verifiable e-voting scheme without tallying authorities. In *ESORICS (2)*, volume 9879 of *Lecture Notes in Computer Science*, pages 223–240. Springer, 2016.
23. D. R. Stinson. *Cryptography: theory and practice*. Chapman and Hall/CRC, 2005.