Features and usage of Belenios in 2022

Véronique Cortier, Pierrick Gaudry, Stéphane Glondu

Features and usage of Belenios in 2022

Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu

Université de Lorraine, CNRS, Inria, France

Abstract. Belenios is an open-source Internet voting protocol associated to a free voting platform, launched in 2015. A detailed overview of the protocol has been presented in [6] in 2019 and its complete, up-to-date specification is public [7]. Since 2019, the use of Belenios has significantly increased with more than 1,400 elections organized each year in 2020 and 2021, and a total of more than 100,000 received ballots.
We report here on the new features added to Belenios since 2019 that include weighted votes, flexible counting methods (*e.g.* Condorcet or STV) thanks to mixnets, and crowdsourced translation with the support of more than 10 languages. Moreover, we have improved the auditability of Belenios in practice, both for voters and authorities.

1 Overview of Belenios

Belenios [5] has been originally inspired by Helios [1]. Compared to Helios, it involves an additional authority (credential authority) to prevent a dishonest server from ballot stuffing. Many cryptographic features have been added since then, such as threshold decryption, mixnets, and blank votes. We briefly survey here the general behaviour of Belenios.

Belenios includes four actors. The **voters** (and their voting devices), the **voting server**, the **decryption authorities**, the **credential authority (CA)**, the **administrator**, and **auditors**. The administrator has no cryptographic role but she is in charge of entering the voter list (a list of email addresses), managing the authorities, defining the start and end date of the election.

Setup. CA generates and sends a private credential to each voter, typically by email. It also sends the list of the associated public credentials to the voting server. The decryption authorities jointly compute the public key of the election.

Voting phase. A voter enters her credential and selects her candidates(s) on her voting device. Her voting choice is encrypted using an homomorphic encryption scheme (ElGamal) and signed with the credential. The signed and encrypted ballot is sent to the voting server once the voter has authenticated herself to the server, using a one-time password sent by email (other authentication mechanisms are supported). The need for both password and credential prevents from ballot stuffing unless the voting server and the credential authority collude. The voter can check on the (public) ballot box that her ballot is present.

Tally. Thanks to the homomorphic property, anyone can compute a ciphertext that contains the number of received votes, for each candidate. The decryption authorities jointly decrypt this ciphertext and produce a proof of correct decryption.

Security properties. The security of Belenios has been studied in several papers [5,4,2]. It preserves **vote secrecy** as long as a threshold of authorities are honest. More precisely, internally, Belenios supports any conjunction of thresholds k_i out of n_i of authorities; on our voting platform, we therefore require the need for the private key of the server and of a threshold of k out of n external decryption authorities, chosen by the electoral board. This protects against an administrator who would silently remove the server as trustee and impersonate all external decryption trustees, as it could be the case in Helios, if participants do not closely supervise who are the decryption authorities.

Belenios is **verifiable** in the following sense. Anyone can check that the result corresponds to the ballots on the bulletin board, thanks to the zero-knowledge proofs (*universal verifiability*). Anyone can check that ballots are encryption of valid candidates and have been produced by legitimate voters, assuming that either the server or the registrar is honest (*eligibility*). Voters can check that their ballot is on the bulletin board (*individual verifiability*).

Belenios however does not guarantee *cast-as-intended*: a malicious voting device could encrypt a candidate different from the choice of the voter. We could easily add the Benaloh challenge [3]. However, several studies [9] have shown that it is very hard to get right in practice and, when badly used, it may even leak the voter's vote. We hope to add a more practical cast-as-intended mechanism in the future.

2 New features

Belenios includes several recent features that, to our knowledge, are not available on other open (and secure) Internet voting platforms.

Weighted votes. A repeated request from our users was to offer weighted votes, where voters may have a different weight. For example, in some sport associations, a voter id may have a number of votes w_{id} that depends on the size of her club. Thanks to the homomorphic property of encryption, it is easy to combine the ballots b_{id} with their weights by computing $\prod_{id} b_{id}^{w_{id}}$ before decryption. The rest of the protocol remains unchanged. Of course, auditors should inspect the voting list even more carefully to check that voters have the expected weight.

Alternative voting. When homomorphic encryption is used, a voter selects between k_1 and k_2 candidates, among a list of n candidates, or vote

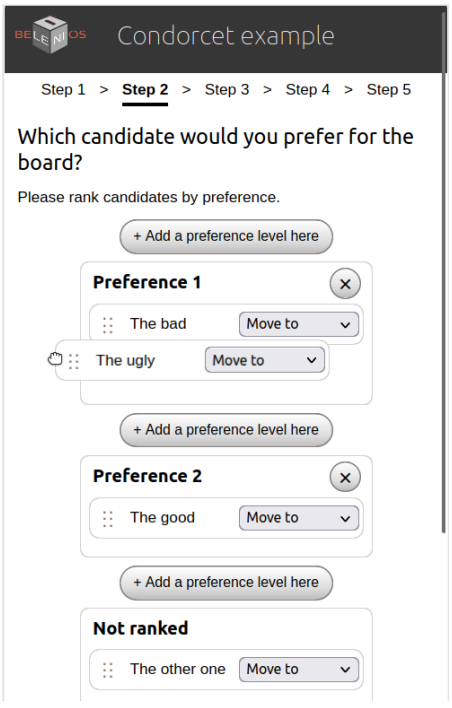

Fig. 1. Interface for preferential voting.

blank (if allowed). Mixnets have been implemented in Belenios so that alternative voting methods can be used, such as STV, Condorcet, or Majority Judgement. We have used the verifiable mixnets proposed in [8]. Despite the additional complexity of mixnets, decryption authorities can still play their role through their browser (our Javascript code takes about 5 minutes for shuffling 1000 ballots, and this grows linearly with the number of voters). We have also adapted the voting interface to support alternative voting as illustrated in Figure 1

Multiple languages. Belenios is used in several countries, well beyond the academic community, thanks to the fact that the voting platform is available in about 12 languages (Czech, English, French, German, Greek, Italian, Norwegian, Polish, Portuguese, Romanian, Spanish, Ukrainian). New languages can easily be added and translations may be amended by any volunteer, thanks to the Weblate platform, available at `https://hosted.weblate.org/projects/belenios/`.

3 Auditability

Many academic voting protocols are verifiable: the authorities as well as any external observer can monitor the ballot box and check that the ballots are well formed and that the result corresponds to the ballots. However, it is not that easy to have authorities who verify in practice since they do not have all the ability to run specific software. Therefore, in Belenios, the main page of an election includes a part, as illustrated in Figure 2, that displays several cryptographic elements (hashes). It allows decryption authorities to check easily (without any software) that their public keys are indeed used in the election and similarly for the credential authority.

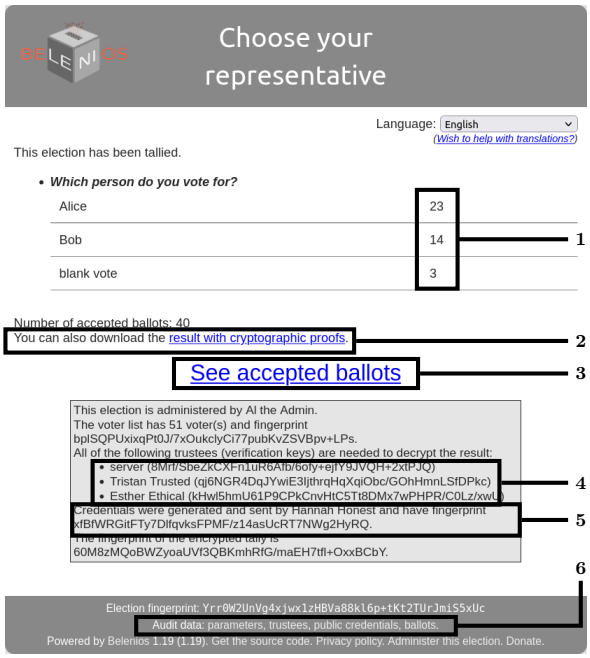

Fig. 2. Final page, including audit data. **1.** Results in human-readable form. **2.** Link to cryptographical proofs. **3.** Link to human-readable ballot box. **4.** Decryption authorities and fingerprints of their keys. **5.** Credential authority and fingerprints of their public parts. **6.** Links to complete machine-readable audit data.

Then a program automatically checks that the hashes displayed on the election page indeed correspond to the election data and that all the cryptographic checks are valid (e.g. validity of the signatures and zero-knowledge proofs). This program also makes sure that no ballot is removed. It can be run by any auditor.

Moreover, for usability reasons, voters vote using a Javascript downloaded from the server. The authorities similarly perform their operations through a Javascript. An auditor should check that these javascripts are indeed the genuine ones. To ease this audit, the pages served by the server have been made constant (this task of making all pages constant is not fully finished yet).

Importantly, the detailed audit procedure, for each actor of the protocol (including voters, authorities, and the administrator) is specified precisely on the Belenios website at `https://www.belenios.org/instructions.html`.

4 Usage of Belenios

Everyone is welcome to deploy their own Belenios server, fitting their technical or legal needs. We are aware of two dozens of such external deployments, because the persons in charge asked us for some help or advice. The only precise statistics we can do is for our own public platform, for which we report the monthly number of elections and number of voters on the Belenios public platform (see Figure 3). The effect of Covid-19 lockdowns is visible, but a good share of users who started using Belenios on this occasion continued thereafter. A seasonal effect is visible: less elections are run during Summer break. Typical users of our platform are academics and associations. Belenios is also used by a German political party and some EU institutions.

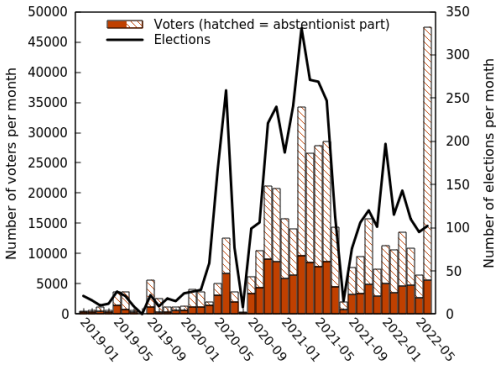

Fig. 3. Usage of the public platform.

References

1. B. Adida. Helios: Web-based open-audit voting. *USENIX'08*, pp 335–348, 2008.
2. S. Baloglu, S. Bursuc, S. Mauw, and J. Pang. Election verifiability revisited: Automated security proofs and attacks on Helios and Belenios. *CSF'21*, pp 1–15, 2021.
3. J. Benaloh. *Verifiable secret-ballot elections*. PhD thesis, Yale University, 1987.
4. V. Cortier, C. Dragan, P.-Y. Strub, F. Dupressoir, and B. Warinschi. Machine-checked proofs for electronic voting: privacy and verifiability for Belenios. *CSF'18*, pp 298–312, 2018.
5. V. Cortier, D. Galindo, S. Glondu, and M. Izabachene. Election verifiability for Helios under weaker trust assumptions. *ESORICS'14*, pp 327–344. Springer, 2014.
6. V. Cortier, P. Gaudry, and S. Glondu. *Belenios: A Simple Private and Verifiable Electronic Voting System*, pp 214–238. Springer, 2019.
7. S. Glondu. Belenios specification - version 1.19. `http://www.belenios.org/specification.pdf`, 2022.
8. R. Haenni, P. Locher, R. Koenig, and E. Dubuis. Pseudo-code algorithms for verifiable re-encryption mix-nets. *FC'17*, pp 370–384. Springer, 2017.
9. K. Marky, O. Kulyk, K. Renaud, and M. Volkamer. What did I really vote for? *ACM CHI'18*, 2018.