H. Langweg, M. Meier, B.C. Witt, D. Reinhardt (Hrsg.): Sicherheit 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 197

# Comparative Usability Evaluation of Cast-as-Intended Verification Approaches in Internet Voting

Karola Marky[1], Oksana Kulyk[2], Melanie Volkamer[3 4]

**Abstract:**

Internet Voting promises benefits like the support for voters from abroad and an overall improved accessibility. But it is accompanied by security risks like the manipulation of votes by malware. Enabling the voters to verify that their voting device casts their intended votes is a possible solution to address such a manipulation - the so-called *cast-as-intended verifiability*. Several different approaches for providing cast-as-intended verifiability have been proposed or put into practice. Each approach makes various assumptions about the voters' capabilities that are required in order to provide cast-as-intended verifiability. In this paper we investigate these assumptions of four chosen cast-as-intended approaches and report the impact if those are violated. Our findings indicate that the assumptions of cast-as-intended approaches (e.g. voters being capable of comparing long strings) have an impact on the security of Internet Voting systems. We discuss this impact and provide recommendations how to address the identified assumptions and give important directions in future research on usable and verifiable Internet Voting systems.

**Keywords:** E-Voting; Cast-as-Intended Verifiability; Usability Evaluation

## 1 Introduction

Internet Voting can offer many benefits. Voters from abroad are easily integrated into an election and the election overall has a better accessibility. Although these benefits can be appealing for governments and private organizations planning to introduce Internet Voting, it also introduces new risks that are not present in traditional paper-based voting. One of those risks lies in the manipulation of votes during vote casting. The votes are cast in an unsupervised environment, in which the voters use their own personal devices. Malware on the voters' personal devices cannot be ruled out.

Hence, if an adversary wants to maliciously influence the election outcome, he or she could take control over the voting devices via malware. This malware could replace votes cast by the voters with votes for the adversary's preferred candidate. Enabling the voters to

[1] Technische Universität Darmstadt, Deutschland, karola.marky@secuso.org
[2] Technische Universität Darmstadt, Deutschland, oksana.kulyk@secuso.org
[3] Karlsruhe Institute of Technology, Deutschland, melanie.volkamer@kit.edu
[4] Technische Universität Darmstadt, Deutschland

doi:10.18420/sicherheit2018_15

verify that their voting device casts the intended vote is a possible solution to address such a manipulation - the so-called *cast-as-intended verifiability*. The concept of cast-as-intended verifiability is an aspect of *end-to-end verifiability*[5] which is the possibility to verify the integrity of the election result through all stages of an election. However, as opposed to other kinds of verification encompassed by end-to-end verifiability, existing cast-as-intended approaches require active voter involvement. Therefore, usability plays an important role in ensuring that the verification can be carried out successfully. An important aspect of usability is ensuring that the voters possess the necessary capabilities required to carry out the verification process. Assumptions on such voter capabilities are often implicitly made by the designers of cast-as-intended approaches. These assumptions and their impact on the cast-as-intended verification have, to our knowledge, not been systematically investigated yet.

In this paper we investigate the assumptions on voter capabilities of four cast-as-intended approaches in an expert evaluation. We identify steps that require voter interaction and give a detailed explanation of each investigated approach from the voter's perspective. For each step we report the assumptions about voter capabilities as well as the impact if the assumptions are violated. Our investigation indicates that the violation of assumptions can be exploited by adversaries to manipulate the election and therefore compromise the Internet Voting system's security. Furthermore, our investigation identifies security-critical assumptions on voter capabilities that cannot be expected to hold among the general voters population. Such capabilities include voters being able to compare long random sequences of characters or having access to multiple computational devices. It follows that the capabilities of voters should not be underestimated and should be taken into account when choosing and/or developing an Internet Voting system. Therefore, our work is a stepping stone into a holistic evaluation and comparison of different cast-as-intended approaches.

## 2 Evaluation Methodology

Several attempts to classify available cast-as-intended verification approaches have been made in the literature [GC16, Pu17, KW17]. The classification by Guasch [GC16] is the most complete available in the literature and considers the following five categories: (1) *challenge or cast*, (2) *decryption-based*, (3) *verifiable optical scanning*, (4) *verification with codes* and (5) *hardware-based verification*. For our evaluation we decided to investigate one approach per category. The category *verifiable optical scanning* does not contain approaches for Internet Voting and was therefore dropped.

The goal of our work was to evaluate the usability of the approaches independent of the user-interface. While implementing the proposed approaches and conducting usability studies

[5] The other parts are recorded-as-cast (the recorded vote does not differ from the cast vote, tallied-as-recorded (the recorded vote is correctly included in the tally) and eligibility verifiability (only votes cast by eligible voters are included in the tally) [Ad06, GV10].

is a more traditional approach, it also has drawbacks. The choice of the user-interface has an impact on the usability of the solution. In an election the user-interface is often adjusted to the specific electorate and election setting. While the question of developing the most usable user-interface for a given verifiability method is important, it hampers evaluating and comparing the usability of the approaches themselves. The approaches play an important role in usability, in case they make assumptions on voter capabilities (e.g. by requiring the voter to compare character sequences of a given length). Hence, a user-interface-independent evaluation is a vital step in deciding whether a verifiability approach is usable enough to be used in an election. Therefore, we also do not cover user-interface-based assumptions (e.g. the visibility and position of buttons or the understandability of instructions), because of its dependence from the specific user-interface.

Our method is based on the so-called *cognitive walkthrough* [Po92]. The original method targets the evaluation of user-interfaces. We adjusted the method to carry out the evaluation independent from the user-interface. We considered the steps that require interaction from the voters attempting to verify their votes (i.e. that cannot be automated by delegating them to a voting client or supporting software). Based on these steps we performed a modified cognitive walkthrough with the following questions: (a) Which assumption about voter capabilities is made? and (b) What is the impact if the assumption does not hold? If available, we relied on the literature in the field human-computer interaction that provides insight of user capabilities to perform a given task. The process was conducted by two authors of the paper independently from each other. The findings were discussed and a final assumption and impact for each step of the approach was agreed upon.

## 3 Results

During our evaluation we collected the following assumptions on voter capabilities in the investigated approaches. We give an overview on the assumptions first and subsequently describe the approaches. The assumptions made by the investigated approaches are:

**A1: Entering.** The voter enters a value without errors.
**A2: Storing.** The voter stores a value without errors.
**A3: QR Scan.** The voter is capable of scanning a QR code.
**A4: Device.** The voter has access to all devices required to carry out verification.
**A5: Comparison.** The voter is able to perform comparisons without errors.
**A6: Access.** The voter can access the component that publishes data required for verification.
**A7: Search.** The voter is able to search for a value (e.g. on a bulletin board).

In the evaluation we use two terms for describing the impact of an assumption. A *false positive* denotes that the voter uncovers an error or manipulation that in reality is not present. This might result in a distrust in the Internet Voting system. Although everything functioned correctly, the voters might think they have uncovered errors and might therefore loose confidence in the Internet Voting system. A possible consequence is the non-usage of

| Assumption | Helios | Estonian System | Neuchâtel (2015) | Du-Vote |
|---|---|---|---|---|
| A1 (Entering) | - | - | ✓ (Code) | ✓(Code) |
| A2 (Storing) | - | - | - | ✓ (Code) |
| A3 (QR Scan) | ✓ | ✓ | - | - |
| A4 (Device) | ✓ | ✓ | - | ✓ |
| A5 (Comparison) | ✓ (Selection) | ✓ (Selection) | ✓ (Code) | ✓ (Code) |
| A6 (Access) | - | - | - | ✓ |
| A7 (Search) | - | - | - | ✓ (Code) |

Tab. 1: Overview of assumptions by approach. ✓ denotes required, - denotes not required.

the Internet Voting system in the future. A *false negative* denotes that the voter does not uncover an error or manipulation that is present. In this case manipulation that took place is not uncovered. Therefore, the integrity of the tally result cannot be assured. Adversaries willing to maliciously influence the election result might exploit usability problems of the verification approaches and attack voters deliberately. Both cases constitute severe problems in an Internet Voting system. The second case influences the security showing that usability has an impact on it. In the first case, although security is given, it is ineffective in convincing the voters. An overview of the assumptions taken by the investigated approaches is given by Tab. 1.

### 3.1 Helios (Challenge or Cast)

Helios [Ad08] utilizes the so-called Benaloh Challenge [Be06, Be07] for cast-as-intended verification. Several proposals for the Benaloh Challenge that differ regarding voter interaction and security are available in the literature [Ad08, Ka11b, Ka11a, Ne14]. For our analysis we choose the approach by Neumann *et al.* [Ne14]. The approach is specifically designed to support the usage of different devices for voting and verifying. The voters are required to use a second device (e.g. a smartphone) for verifying. Therefore, it mitigates the assumption of a trusted voting device[6].

The Benaloh challenge enables verification that the voting client acts correctly via a so-called challenge or cast approach. After encrypting their selections, the voters have two options: (1) they can cast their encrypted vote or (2) they challenge the voting client by verifying whether the encrypted vote contains the intended selection. At the time of preparing an encrypted vote, a potential adversary controlling the voting client does not obtain knowledge, whether the encrypted vote will be cast or challenged. Therefore an adversary cannot be certain whether a manipulation will be undetected. We further describe the Benaloh Challenge in more details.

[6] The original Helios approach [Ad08] does neither exclude nor enforce the usage of different devices, but it is not specifically designed to support the transfer of verification data to a different device.

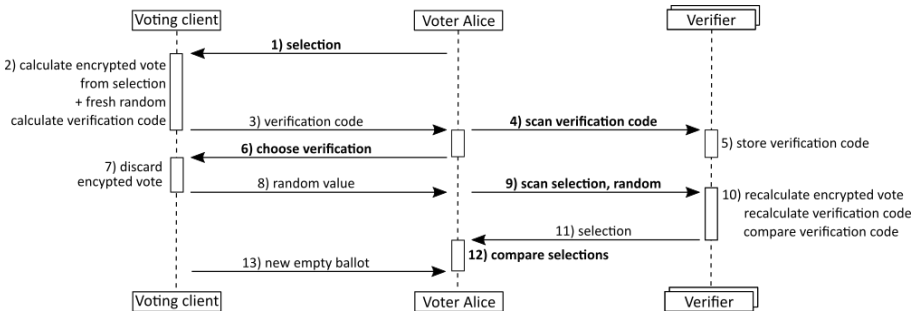


Fig. 1: Sequence diagram of one Benaloh Challenge. Steps with voter interaction are bold.

Fig. 1 provides a sequence diagram of the Benaloh Challenge with a verification device. The voter Alice commences by selecting a voting option in the voting client. The voting client encrypts Alice's selection probabilistically and derives a verification code, which is a hash from the resulting encrypted vote. Then the voting client displays this verification code to Alice. The verification code serves as a commitment of the voting client to the encrypted vote. The verification code is also displayed as QR code which Alice scans with a verification device (e.g. smartphone). Therefore, Alice needs access to such a device (A4). Otherwise, she would not able to verify at all. An adversary might be able to obtain knowledge which voters do not have access to a verification device and manipulate their votes without detection.

Alice decides whether to cast or to verify the encrypted vote. In case of verification, the voting client displays the verification data to her as a QR code. This data consists of the selection Alice previously made and the random value used during encryption. Alice scans the QR code with the verification device. Therefore, she has to be able to perform the scan (A3), otherwise she cannot verify which results in the same consequences that Alice would face by not having access to a verification device. The verification device runs an alternative software - the so-called *verifier* - to recalculate the encrypted vote. Then it derives a verification code from the encrypted vote and automatically compares it to the previously scanned one. If the verification codes match, the selection used for computation is displayed to Alice. She compares this to her previous selection. If both match, the encrypted vote contained Alice's intent, otherwise Alice can trigger an alarm. Alice is assumed to perform the comparison without errors (A5). Errors would result in a false positive (uncovering a manipulation that is not present). After verification Alice has to start from the beginning with an empty ballot[7]. If Alice chooses to cast her vote she still has to take note of the verification code by scanning the corresponding QR code. After casting, Alice's verification code is published on the election's bulletin board. In order to verify that the correct vote was cast, the verification device looks up Alice's verification code on the bulletin board. Alice is informed whether her verification code is on the bulletin board.

[7] The random value could be used for breaking Alice's vote secrecy in case she would cast this vote.

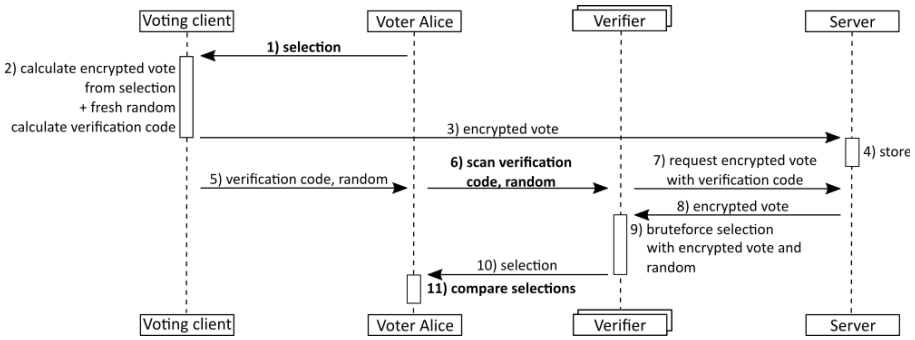


Fig. 2: Sequence diagram of verification in the Estonian system. Steps with voter interaction are bold.

### 3.2 Estonian System (Decryption-based)

The Estonian voting system [HW14] is a decryption-based approach. This means that the ciphertext of the encrypted vote is investigated in order to perform cast-as-intended verification. Because the ciphertext of the cast encrypted vote is inspected, vote secrecy is broken. Fig. 2 provides a sequence diagram of cast-as-intended verification in the Estonian system.

Alice opens the voting client and authenticates via an ID card. She makes her selection which is encrypted probabilistically by the voting client and subsequently send to the vote storage server. For cast-as-intended verification, the voting client generates a QR-code which contains the random value used during encryption and a hash of the encrypted vote that we denote as verification code. To verify Alice needs to install a verification application on her mobile device and scan the QR code to transfer the random value and the verification code. Therefore, Alice requires access to a mobile device (A4), otherwise she cannot verify. Alice has to be able to scan the QR code (A3), otherwise Alice cannot execute verification at all. The same consequences mentioned above in the Helios description apply here. An adversary might be able to obtain knowledge whether the voter has access to a verification device and can successfully manipulate a vote in case the voter does not have access.

The verification app uses the verification code to download the encrypted vote from the vote storage server. Using the random value, the verification app bruteforces the content of the encrypted vote by encrypting all possible voting options and comparing the resulting encrypted votes with the one downloaded from the server. Then it displays the voting option that matches the encrypted vote of Alice. She has to compare this to her previous selection without errors (A5).

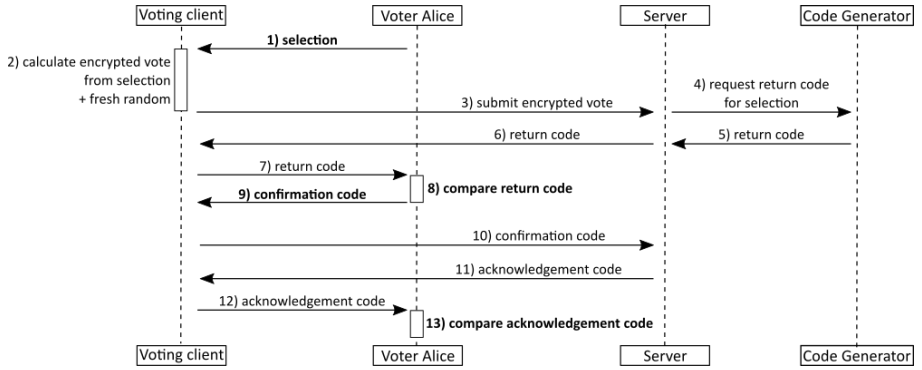


Fig. 3: Sequence diagram of verification in Neuchâtel (2015). Steps with voter interaction are bold.

### 3.3 Neuchâtel (2015) (Verification with Return Codes)

The cast-as-intended approach of Neuchâtel (2015) [GGP15] is based on return codes (see Fig. 3). Alice opens the voting client and indicates her selection. The selection is encrypted and sent to the voting server. The voting server contacts the code generator which answers with the return code (here called verification code) corresponding to Alice's selection. The verification code is forwarded to Alice. Prior to the election Alice has received a code sheet (e.g. via postal mail) listing her combinations of voting options and verification codes. Alice checks, whether the displayed verification code belongs to her selection. To successfully verify, she is assumed to compare the received verification code to the one on her code sheet without errors (A5). Comparing the received verification code to the wrong code on the code sheet might result in a false positive. Making an error during the comparison, even if Alice compares to the correct verification code on the code sheet, also leads to a false positive.

If the verification code is correct, Alice sends a confirmation code, that she also finds on her code sheet, to the voting server. She assumed to enter it without errors (A1), if she makes an error she cannot confirm that she compared the codes and her vote will not be included in the tally. The voting server finally responds with an acknowledgement code to confirm that it received the confirmation code. Alice is assumed to compare the received acknowledgement code without errors (A5). If she makes an error during comparison, this results in a false positive.

### 3.4 Du-Vote (Hardware Token)

In the Du-Vote approach [Gr15] cast-as-intended verification is based on the access to a trusted device for performing specific computations. In the case of Du-Vote the trusted device is a hardware token that Alice needs access to (A4). Otherwise she can neither vote

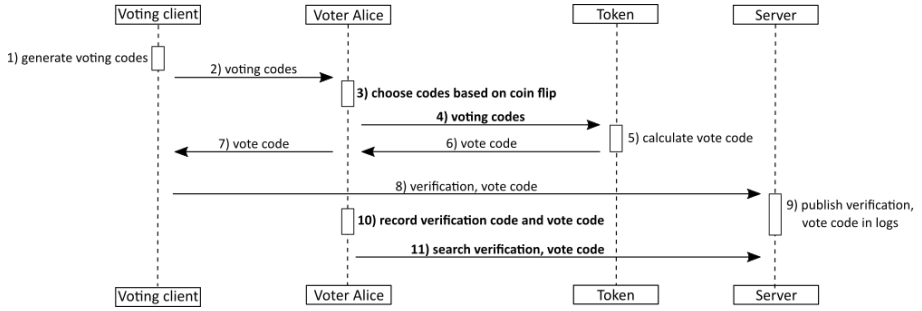


Fig. 4: Sequence diagram of verification in Du-Vote. Steps with voter interaction are bold.

nor verify. Fig. 4 provides a sequence diagram of Du-Vote. Alice gets a hardware token with unique embedded keys. Those keys are registered on the voting server. To vote Alice has to authenticate herself by previously received credentials. The voting client calculates two random values per voting option. The random values are grouped in two sets. Alice is instructed to flip a coin to randomly select one set. Now she enters all random values of the selected set into the hardware token. Then she enters the random value from the non-selected set that belongs to her selected voting option. The hardware token generates a vote code, that Alice enters in the voting client. The correctness of the verification is based on the assumption that Alice generates a correct vote code by using the hardware token. This is only the case if Alice does not make errors while entering the random values (A1). If only one wrong digit is entered, this fault propagates and leads Alice to a false negative.

Alice records her vote code as well as the vote identifier, which was generated by the voting client. To provide verification, the voting server logs all votes and publishes this data. Alice searches these logs for her vote identifier and vote code. If they are present in the log, Alice's vote was cast-as-intended. Alice has to be able to find and open the server logs (A6). If Alice cannot perform this step, she cannot verify her vote. It also is assumed that she can search these logs for her vote identifier and vote code (A7). This means that it is assumed that Alice has previously recorded these values without errors (A2). If Alice did not record these values, she cannot verify. If she made an error during recording, she will not find her vote in the log and therefore assume an error (false positive). If Alice is not able to find her verification code in server logs although it is present, she will also assume an error. Finally, Alice has to be able to compare her vote code without errors (A5).

## 4 Discussion

In the modified cognitive walkthrough we determined seven assumptions that are made by the approaches and described the impact if these assumptions are violated. In this section we discuss the identified assumptions, the investigated approaches in context of the

assumptions they rely on, and interconnections of security and usability of Internet Voting systems in context of assumptions for cast-as-intended verification.

**Assumptions.** Assumption A5 (Comparison) is made by all investigated approaches. Therefore, the task of comparing verification codes or other data (e.g. the voter's selection) is of crucial importance for verifying successfully. The verification codes are random character sequences, the voter's selection is ordinary text (e.g. the name of a political party). Hence, there is a difference in the difficulty of comparing. Comparing random character sequences, which is required in Neuchâtel (2015) and Du-Vote, is considered more difficult. The human short time memory is limited to chunks of seven [Mi56, Ta17], therefore, verification codes with more than seven characters are difficult to process for the voters. Hence, the visualization of verification codes should be adjusted (e.g. splitting the codes in chunks [Ta17]), to support an error-free comparison. The assumptions A2 (Storing) and A7 (Search) are similar to the comparison task (A5). If a voter has to store a verification code by writing it down or search for it manually, the same aspects apply.

**Approaches.** Neuchâtel (2015) requires the least assumptions. The voter has to enter the confirmation code (A1) and make two comparisons (A5) (return code and acknowledgement code). Therefore, from a usability perspective Neuchâtel (2015) is the most promising of the investigated approaches. The scalability of Neuchâtel (2015) and other return code-based approaches, however, appears questionable. One return code per voting option is required and its length depends on the total number of voters. Therefore, elections with many voting options and voters as well as complex elections might be problematic. The investigation and improvement of the scalability of return code-based approaches forms an important task of future work. The Du-Vote approach requires the most assumptions. Among them the access to a trusted hardware token. From a usability perspective Du-Vote is the least promising approach. The assumptions A2 (Storing) and A5 (Comparison) could be removed by providing a device that supports the voter. Therefore, at least A4 would have to be added.

**Security and Usability.** The usability of the cast-as-intended verification is crucial for the security of the Internet Voting system, but other aspects of security should be considered as well. All investigated approaches rely on different security-related assumptions for ensuring the integrity of the election result. An example is the trustworthiness of the verification devices. Helios, the Estonian System and Du-Vote fail to provide verifiability unless the voter has access to a trustworthy verification device or hardware token (A4). This might be difficult to ensure if the devices in the voter's possession (i.e. a personal computer and a smartphone) are synchronized. Malware infection of one device can also affect the other device (see e.g. [KvdVB16] for elaborating on such vulnerabilities). Neuchâtel (2015) does not require such trusted devices, but requires auxiliary materials. This auxiliary material is a code sheet that is assumed to be generated and distributed in a trustworthy

manner. While the differences in the security models of the investigated approaches are out of scope of this paper, they have to be investigated in conjunction with usability-related assumptions in order to provide a holistic evaluation and comparison of the Internet Voting systems. Another example of the interconnection between the usability of cast-as-intended approaches and the security of the Internet Voting system is the attempt to reduce burden on the voter by automating steps of the verification process. A prominent example is the comparison that the voters have to perform (A5). The comparison could be automated via a supporting software [Ta17]. Automation aims to simplify the verification process and to increase the usability of the cast-as-intended approaches. However, it also introduces new security-related assumptions. Such an assumption would be trust in the supporting software performing the comparison. Such a trade-off between the assumptions should therefore be further investigated in each individual election setting.

## 5 Related Work

The usability of cast-as-intended approaches in Internet Voting has been investigated in few works so far [WH09, Ka11b, FR12, Ac14, Re17]. Several works [WH09, Ka11b, Ac14, Re17] focus on the usability of the Benaloh Challenge [Be06, Be07] which is used in Helios [Ad08]. The usability of Internet Voting systems that provide return code-based verification was evaluated in [FR12, Ku17]. While these studies provide valuable insights into the usability of the cast-as-intended verification, they focus on evaluating the usability of specific implemented Internet Voting systems. The required voter interaction, however, is given by the process that a voter has to follow in order to verify which is user-interface-independent.

## 6 Summary and Future Work

In this paper we investigate four approaches for providing for cast-as-intended verifiability. The approaches provide the possibility for the voters to verify that their true intents are represented electronically. Each approach makes assumptions regarding voter capabilities. In our investigation we show that assumptions on voter capabilities have an impact on the Internet Voting system's security and on the integrity of the election result. We identified Neuchâtel (2015) as most promising approach from the usability perspective. Helios and the Estonian system form a middle ground which is followed by the Du-Vote approach. This is a stepping stone into a holistic evaluation and comparison of different cast-as-intended approaches.

Usability is not the only aspect with an impact on the security of the Internet Voting system. There is a tradeoff of usability, security and potentially other influence variables in the scope of cast-as-intended verification. The investigation of this tradeoff forms an important task of future work.

## References

[Ac14] Acemyan, C. Z.; Kortum, P.; Byrne, M. D.; Wallach, Dan S.: Usability of Voter Verifiable, End-to-End Voting Systems: Baseline Data for Helios, Prêt à Voter, and Scantegrity II. In: The USENIX Journal of Election Technology and Systems (JETS), 2(3). USENIX Association, pp. 26–56, 2014.

[Ad06] Adida, B.: Advances in Cryptographic Voting Systems. PhD thesis, Massachusetts Institute of Technology, 2006.

[Ad08] Adida, B.: Helios: Web-based Open-Audit Voting. In: Proceedings of the 17th USENIX Security Symposium. USENIX Association, pp. 335–348, 2008.

[Be06] Benaloh, J.: Simple Verifiable Elections. In: Proceedings of the 1st Electronic Voting Technology Workshop (EVT). USENIX Association, 2006.

[Be07] Benaloh, J.: Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. In: In Proceedings of the 2nd Electronic Voting Technology Workshop (EVT). USENIX Association, 2007.

[FR12] Fuglerud, K. S.; Røssvoll, T. H.: An Evaluation of Web-Based Voting Usability and Accessibility. In: Universal Access in the Information Society (UAIS), 11(4). Springer, pp. 359–373, 2012.

[GC16] Guasch Castelló, S.: Individual Verifiability in Electronic Voting. PhD thesis, Universitat Politécnica de Catalunya, 2016.

[GGP15] Galindo, D.; Guasch, S.; Puiggalí, J.: 2015 Neuchâtel's Cast-as-Intended Verification Mechanism. In: Proceedings of the International Conference on E-Voting and Identity (VoteID). Springer, pp. 3–18, 2015.

[Gr15] Grewal, G. S.; Ryan, M. D.; Chen, L.; Clarkson, M. R.: Du-Vote: Remote Electronic Voting with Untrusted Computers. In: Proceedings of the 28th Computer Security Foundations Symposium (CSF). IEEE, pp. 155–169, 2015.

[GV10] Gharadaghy, R.; Volkamer, M.: Verifiability in Electronic Voting-Explanations for Non Security Experts. In: International Conference on Electronic Voting (EVOTE). Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, pp. 151–162, 2010.

[HW14] Heiberg, S.; Willemson, J.: Verifiable Internet Voting in Estonia. In: Proceedings of the International Conference on Electronic Voting: Verifying the Vote (EVOTE). IEEE, pp. 1–8, 2014.

[Ka11a] Karayumak, F.; Kauer, M.; Olembo, M. M.; Volk, T.; Volkamer, M.: User Study of the Improved Helios Voting System Interfaces. In: Proceedings of the 1st Workshop on Socio-Technical Aspects in Security and Trust (STAST). IEEE, pp. 37–44, 2011.

[Ka11b] Karayumak, F.; Olembo, M. M.; Kauer, M.; Volkamer, M.: Usability Analysis of Helios-An Open Source Verifiable Remote Electronic Voting System. In: Proceedings of the Electronic Voting Technology Workshop/ Workshop on Trustworthy Elections (EVT/WOTE). USENIX Association, 2011.

[Ku17] Kulyk, O.; Neumann, S.; Budurushi, J.; Volkamer, M.: Nothing Comes for Free: How Much Usability Can You Sacrifice for Security? In: IEEE Security & Privacy, 15(3). IEEE, pp. 24–29, 2017.

[KvdVB16] Konoth, R. K.; van der Veen, V.; Bos, H.: How Anywhere Computing Just Killed your Phone-Based Two-Factor Authentication. In: Proceedings of the International Conference on Financial Cryptography and Data Security (FC). Springer, pp. 405–421, 2016.

[KW17] Khazaei, S.; Wikström, D.: Return Code Schemes for Electronic Voting Systems. In: Proceedings of the International Joint Conference on Electronic Voting (E-Vote-ID). Springer, pp. 198–209, 2017.

[Mi56] Miller, G. A.: The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. In: Psychological Review, 63(2). American Psychological Association, pp. 81–97, 1956.

[Ne14] Neumann, S.; Olembo, M. M.; Renaud, K.; Volkamer, M.: Helios Verification: To Alleviate, or to Nominate: Is That the Question, or Shall we Have Both? In: Proceedings of the International Conference on Electronic Government and the Information Systems Perspective (EGOVIS). Springer, pp. 246–260, 2014.

[Po92] Polson, P. G.; Lewis, C.; Rieman, J.; Wharton, C.: Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces. In: International Journal of Man-Machine Studies, 36(5). Elsevier, pp. 741–773, 1992.

[Pu17] Puiggalí, J.; Cucurull, J.; Guasch, S.; Krimmer, R.: Verifiability Experiences in Government Online Voting Systems. In: Proceedings of the International Joint Conference on Electronic Voting (E-Vote-ID). Springer, pp. 248–263, 2017.

[Re17] Realpe-Muñoz, P.; Collazos, C. A.; Hurtado, J.; Granollers, T.; Muñoz-Arteaga, J.; Velasco-Medina, J.: Eye Tracking-Based Behavioral Study of Users Using E-Voting Systems. In: Computer Standards & Interfaces. Elsevier, 55, pp. 182–195, 2017.

[Ta17] Tan, J.; Bauer, L.; Bonneau, J.; Cranor, L. F.; Thomas, J.; Ur, B.: Can Unicorns Help Users Compare Crypto Key Fingerprints? In: Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI). ACM, pp. 3787–3798, 2017.

[WH09] Weber, J.-L.; Hengartner, U.: Usability Study of the Open Audit Voting System Helios. http://www.jannaweber.com/wpcontent/uploads/2009/09/858Helios.pdf, 2009. Accessed: 22-December-2017.