Process Models for Universally Verifiable Elections

Rolf Haenni, Eric Dubuis, Reto E. Koenig, and Philipp Locher

Bern University of Applied Sciences, CH-2501 Biel, Switzerland
{rolf.haenni,eric.dubuis,reto.koenig,philipp.locher}@bfh.ch

Abstract. In this paper, we analyze the process of performing the universal verification of an electronic election. We propose a general model of the election process and define the data flow into the verification process. We also define the purpose and outcome of the verification process and propose some general categories of tests to be performed during the verification. As a guideline for dealing with negative verification outcomes, we propose some general evaluation criteria for assessing the impact and consequences of the encountered problem. Finally, we generalize the proposed process models to the case of hybrid elections, in which multiple voting channels are available simultaneously. The primary target audience of this paper are people in charge of implementing and organizing verifiable elections in practice.

1 Introduction

Universal verifiability is a key concept for making electronic voting systems secure enough for using them in real political elections. It is a counter-measure against all sorts of threads from very powerful adversaries, which for example may try manipulate the election result by taking control over some of the central system components. To prevent such attacks, the system generates some public election data during the election process, which can be used to reconstruct the final election result in a publicly verifiable manner. Independent third parties (auditors) can then be invited to verify the correctness of the election result based on the cryptographic evidence included in the public election data. Provided that the verification has succeeded, one can then conclude that no such attacks have been conducted. By providing this simple functionality, universal verifiability is a very important trust-establishing measure. Its ultimate goal is to convince even the losers of an election to accept the result [7, 11].

1.1 Universal Verifiability in Practice

One of the major challenges of building a universally verifiable election system is to provide verifiability simultaneously with vote secrecy. Many cryptographic protocols have been invented for that purpose. Their main problem is to define the verification process in a way that the correct election result can be reconstructed without explicitly decrypting the submitted encrypted votes. For this, some anonymization mechanism must be applied to the submitted votes to unlink them from the voters. Techniques for solving this problem, for example mix-nets or homomorphic tallying, are well-understood today and widely applied. Practical systems using these technologies have been introduced for both academic and real-world purposes [2–6].

Based on today's generally accepted understanding that verifiability is crucial for electronic elections, countries such as Switzerland and Estonia have decided to update the requirements for their existing e-voting systems. The following quote from the *Federal Chancellery Ordinance on Electronic Voting* (VEleS) underlines this change of paradigm in Switzerland [1, page 3]:

> *"Auditors receive proof that the result has been ascertained correctly. They must evaluate the proof in a observable procedure. To do this, they must use technical aids that are independent of and isolated from the rest of the system."*

To fulfill the extended requirements, the two remaining Swiss e-voting system providers have launched corresponding development projects. By releasing a detailed and comprehensive protocol specification together with two different proof-of-concept implementations [8,9], the CHVote project of the State of Geneva has reached an important milestone in 2017. The launch of the new system, which is currently being developed according to the specification, is planned for the 2019 parliament elections. Similar plans exist for the system offered by the Post CH Ltd, which has officially reached an intermediate expansion stage in early 2018. In both projects, the legal ordinance is clear about implementing proper verification processes along with the introduction of the next-generation systems. However, since VEleS does not further specify the details of such processes, it does not provide sufficient legal grounds for most of the conclusions and recommendations contained this paper.

1.2 Goals and Overview

Despite the recent developments in Switzerland and other places in the world, only little experience exists with respect to conducting an actual verification process for real political elections. The foremost problem is the necessity of providing suitable *technical aids* that offer the desired functionality while satisfying the requirement of being independent from the rest of the system. In some of the above-mentioned systems, such technical aids have never been developed. This leads to a paradoxical situations, where systems are promoted as (potentially) verifiable, but without offering the full package for performing an actual verification.

Another problem is the lack of a common understanding of the exact purpose of a verification and the necessary processes around it. Simple questions like what are the exact input data of a verification process and what are the possible verification results have never been defined in a precise manner. Such a high-level view of the verification process is the main topic of this paper. The goal is to lay the foundations for introducing universal verification into existing and future electoral processes. For this, we look at the commonalities of existing e-voting protocols, propose a high-level summary of the relevant data flow, and finally derive general models for both the election and the verification processes. The paper is written mainly from a technical perspective. Related political, legal, or sociological questions are deliberately left aside.

We will start in Section 2 with a general model of the election process, which defines the principal data flow. This model is general enough to be applicable to both electronic and non-electronic election processes. Based on this model, we propose a definition of the verification process. Particular attention is given to the verification result, which we decompose into five main categories. We also discuss the development process of corresponding verification software. In Section 3, we use the generality of the election process model to define corresponding processes for hybrid voting systems, which provides multiple (electronic and non-electronic) voting channels simultaneously. In this particular setting, additional considerations are necessary to guarantee the completeness of the verification chain. Section 4 summarizes the findings and concludes the paper.

2 Universally Verifiable Elections

An election system's principal function is to establish the correct election result based on the votes submitted by the voters. This should be done in a way that even the losers of the election will accept the result as correct. In a paper-based election system, this functionality is achieved by involving trustworthy people from all parties in the tallying process. In case of observed or suspected irregularities, election authorities can order a re-tally of the votes by independent third parties to remove any existing doubts. In an electronic election system, this is exactly the purpose of conducting a universal verification, but the evidence necessary for inferring the correctness of the result is derived from cryptographic methods rather than human supervision. Irregularities caused by attacks or software bugs can then be detected in a reliable way. The purposes of re-tallying paper votes and universally verifying electronic votes are therefore largely equivalent.

2.1 Election Process

In order to define universal verification more precisely, we must first introduce an abstract model of an election process. To provide compatibility with most existing election protocols, we suppress technical details as far as possible. A common denominator is the *election period*, during which voters can submit their votes. Independently of the exact length of this period, it defines a natural decomposition of the whole election process into three consecutive phases:

$$\text{pre-election phase} \quad \Rightarrow \quad \text{election phase} \quad \Rightarrow \quad \text{post-election phase.}$$

Each phase generates its own part of the *election process data*, which contains the auxiliary cryptographic evidence required to perform the verification of the election result. For the general understanding of the election and verification processes, it is not necessary to further specify the exact content of this data, but it is important to keep in mind that this data is public. Usually, it is written to a public bulletin board, from which it can be retrieved by anyone who wants to perform the verification. The election process data is depicted in Figure 1 as one of the main outputs of the election process.

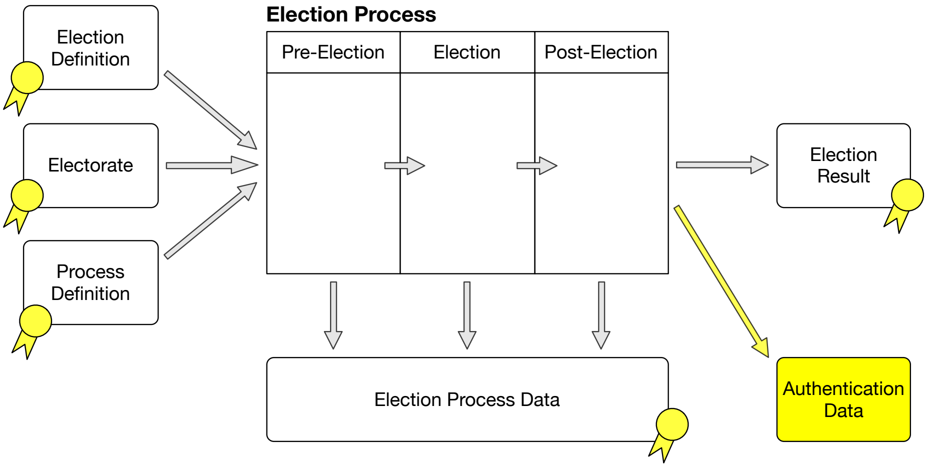


Fig. 1: Abstract model and data flow of an election process.

One of the main inputs of the election process is a document called *election definition*, which defines the details of the election, for example the questions and voting options in a referendum or the list of candidates and election rules in an election. A second input document, which we call *electorate*, contains the list of eligible voters. This document is needed to determine the voter's eligibility and therefore decide about the validity of a submitted vote. Another input document, which we call *process definition*, specifies the details of the election process, for example the start and the end of the election period, but also the identities of the parties and authorities involved in the process or the cryptographic parameters to be used. The party responsible for providing the three input documents is called *election administrator*. Our distinction between election definition and process definition is important in the hybrid setting discussed in Section 3, where a single election definition is combined with two or more process definitions.

The most important output of the election process is the *election result*. We do not further specify the contents of this document, except that we assume that it summarizes the outcome of the election tally, for example by summing up the number of yes/no-votes in a referendum or by simply enumerating all decrypted votes in cleartext. This document represents therefore the official result, which is publicly announced in the aftermath of the election. For this, it is important for this document to contain a signature from the election administrator, which guarantees the correctness of its contents.

The same remark about containing a signature holds for the three input documents and for most parts of the elections process data. We summarize this aspect of the model by assuming an additional output called *authentication data* (yellow-highlighted in Figure 1). In a purely electronic setting, this output will consists of a list of digital signatures with corresponding certificates, from which the authenticity and integrity of all input and output documents can be inferred. As we will see in the next subsection, this aspect defines a particular category of verification steps, which can be performed independently of the rest.

We already mentioned that we kept this process model simple enough for applying it also to the case of non-electronic elections. In that case, some (but not necessarily all) of the involved documents will be paper documents signed by the people that generated them. They may contain declarations that certain manual tasks of the process have been conducted according to the specified procedures. In case of detected irregularities, the existence of such documents can help in identifying the person responsible for causing or overlooking the problem. They are therefore needed for ensuring the plausibility of the election result.

2.2 Verification Process

The process of verifying an electronic election based on the available public data is depicted in Figure 2. The input of the verification process consists of all the public inputs and outputs of the election process model from the previous section. We refer to it as the *election data* and assume that it is available to any person who wants to perform a verification. Note that we consider the election result as part of the election data, which must be checked for correctness. The purpose of the verification process is to perform a series of tests on the election data, which collectively give enough evidence to assess the correctness of the election result. A compilation of the results obtained from performing the necessary tests is what we call the *verification report*. This document is the principal output of the verification process. The software that generates the verification report based on the election data is called *verifier*.

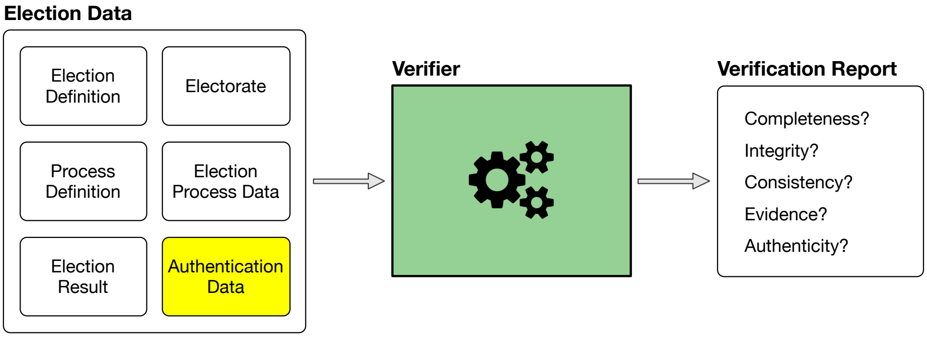


Fig. 2: Abstract model of the verification process.

By defining the verification as a series of individual tests performed by the verifier, it is possible to introduce at least five different top-level categories, according to which the tests can be grouped in a meaningful way (further meaningful categories and sub-categories may exist in more concrete cases). In Table 1, we summarize the meaning of these categories and their differences. One of the purposes of introducing such test categories is to facilitate and systematize the definition of a suitable *test catalog*, which ultimately leads to a fully connected verification chain. This test catalog is the main content of the verifier specification (see Section 2.4). Another purpose of introducing categories is to simplify the organization and presentation of the test results in the verification report.

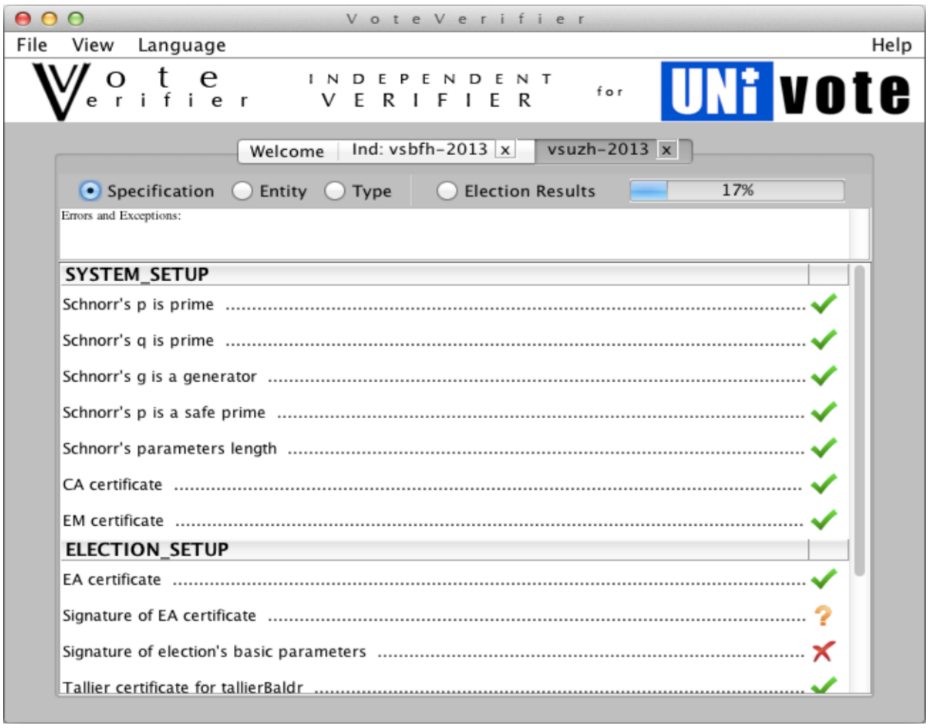
Category	Description
Completeness	Do the available data elements cover the whole election process according to the specification? Do they allow a complete verification chain?
Integrity	Do all data elements correspond to the protocol specification? Are they all within the specified ranges?
Consistency	Are related data elements consistent to each other?
Evidence	Are the cryptographic proofs contained in the election data all valid? Do they provide the necessary evidence to infer the correctness of corresponding protocol steps?
Authenticity	Can the data elements be linked unambiguously to the party authorized to create them?

Table 1: Test categories of the verification process.

An example of a verifier's user interface is depicted Figure 3. It shows the upper part of the verification report for an election at the University of Zürich in 2013 using the UniVote system [4, 10]. The status bar in the upper right corner indicates that the verification is still in progress. The report also shows the results of the first eleven (out of 61) tests. Nine tests succeeded, one test has been dropped due to a missing certificate, and one test failed due to an invalid signature. Assuming that the verifier itself works properly, this indicates that parts of the implemented voting system have not been working properly, or even worse that the election has been exposed to an attack. In any case, it is clear that both the cause and the impact of the exposed problems have to be investigated. Triggering such an investigation in case of irregularities is the main purpose of performing the verification.

2.3 Impact and Consequences of Failed Tests

As illustrated by the above example, using a verifier to conduct the verification of an election can always lead to a situation, in which some tests from the test



Errors and Exceptions:

SYSTEM_SETUP

Schnorr's p is prime ✓

Schnorr's q is prime ✓

Schnorr's g is a generator ✓

Schnorr's p is a safe prime ✓

Schnorr's parameters length ✓

CA certificate ✓

EM certificate ✓

ELECTION_SETUP

EA certificate ✓

Signature of EA certificate ?

Signature of election's basic parameters ✗

Tallier certificate for tallierBaldr ✓

Fig. 3: User interface of the verifier for the UniVote system.

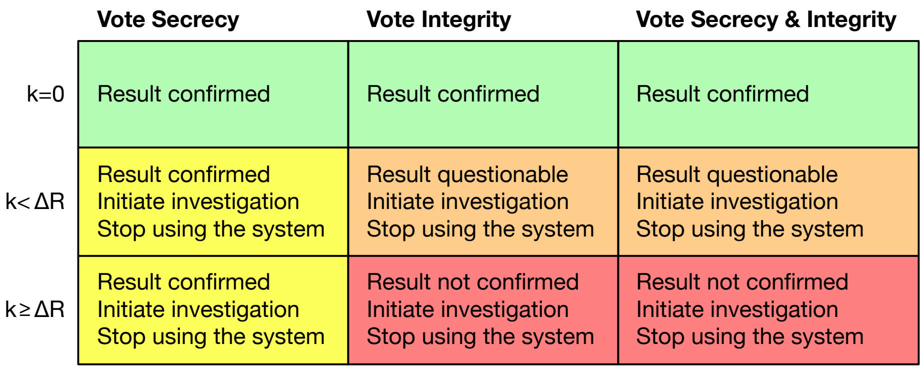
catalog have failed. There are numerous possible causes for a test to fail, but there is presumably no better way for finding the cause than analyzing the particular problem at hand. Giving general recommendations about handling failure cases is therefore quite difficult. Nevertheless, we can at least propose three different evaluation criteria, which may help to classify the impact of the problem and to decide about the next steps.

The first criteria is the maximal number of affected votes. Suppose that N electronic votes have been submitted and that maximally $0 \leq k \leq N$ votes are affected by the problem.[1] Note that this constraint includes the two natural limiting cases of $k = 0$ (no vote affected) and $k = N$ (all votes affected). Another important quantity is the number ΔR of votes, which are necessary to change the winner or the outcome of an election. In a referendum, the general constraint for this number is $1 \leq \Delta R \leq \frac{N}{2}$. For example, for 60 *yes*-votes, 30 *no*-votes, 10 blank votes, and therefore $N = 100$, the outcome could be changed by turning 15 *yes*-votes into *no*-votes. For judging the impact of the problem, it is therefore important to determine if k is smaller or bigger than $\Delta R = 15$. For $1 \leq k < \Delta R$, the impact of the problem may not justify the invalidation of the whole election (similar arguments are used to handle minor irregularities in

[1] In a hybrid election process, both the number of electronic votes and the total number of votes must be taken into consideration.

paper-based elections), but in the more severe case of $k \geq \Delta R$, repeating the whole election can probably not be avoided.

The second criteria refers to the security goal violated by the detected problem. Relative to a single submitted vote, three cases must be distinguished: a violation of the vote's secrecy, a violation of the vote's integrity, or a violation of both the vote's secrecy and integrity.[2] Generally, we consider violations of the vote's integrity to be more critical than violations of the vote's secrecy, because they affect the election results in a direct way. The possible consequences are therefore more drastic in such cases. In Figure 4 we give an overview of the consequences in the scenarios obtained from combining the first two evaluation criteria. It shows for example that vote secrecy violations do not directly invalidate the election result, but that an investigation of the problem's cause is always necessary.



	Vote Secrecy	Vote Integrity	Vote Secrecy & Integrity
k=0	Result confirmed	Result confirmed	Result confirmed
k< ΔR	Result confirmed Initiate investigation Stop using the system	Result questionable Initiate investigation Stop using the system	Result questionable Initiate investigation Stop using the system
k≥ ΔR	Result confirmed Initiate investigation Stop using the system	Result not confirmed Initiate investigation Stop using the system	Result not confirmed Initiate investigation Stop using the system

Fig. 4: Problem scenarios with consequences.

Another important point to consider in case of an unsuccessful verification is the question of whether the problem could possibly be solved by repeating some steps of the election process. For example, the case of a missing or invalid signature could possibly be solved by simply repeating the signature generation. Generally, such recovery procedures mostly exist for data that is not temporarily linked to other parts of the election data. In those cases, only the availability of the data is necessary to conduct the verification, not their moment of creation. Problems encountered with such data can therefore be solved by repeating their creation during a recovery procedure.

Assuming that recovery procedures exist, pursuing them will always be the first choice in case of encountering a problem in the verification report. A general business process model for handling failure cases is depicted in Figure 5. It shows that executing a recovery procedure invokes an additional verification

[2] The main purpose of the universal verification is detecting integrity violations. However, the failing of certain tests can also lead to situations, in which vote secrecy is no longer guaranteed, for example if the signatures of the mixing proofs are all invalid. This could mean that all mixing proofs have been generated by the same party, which can then establish links from cleartext votes to voters.

round. If the problem persists—or if no recovery procedure has existed from the beginning—then an investigation of the problem must be invoked and the result of the investigation must be documented in a report.

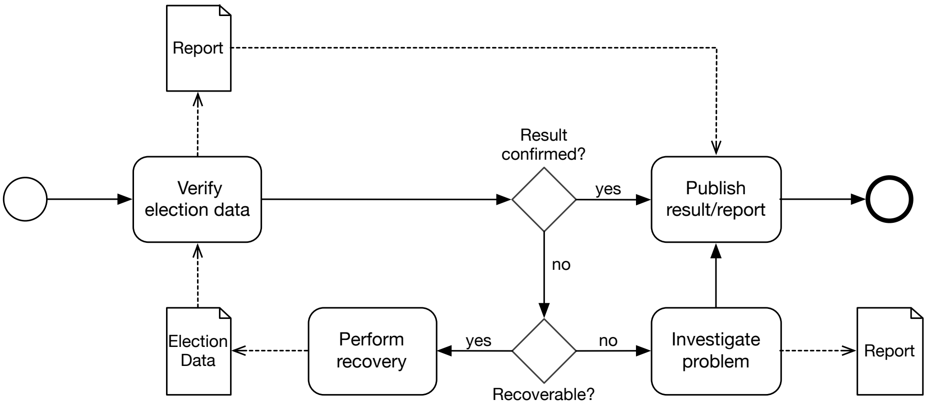


Fig. 5: Process model for handling failure cases and recovering from them.

2.4 Developing the Verifier

The principal technical aid for conducting the verification of an election is the verifier. Clearly, the proper functioning of the verifier is a mandatory precondition for obtaining conclusive verification reports. It is therefore essential that the verifier works exactly in accordance with the specified cryptographic protocol. Any deviation could lead to unpleasant situations in which the verifier reports a failure when everything is correct (false negative) or misses a failure when something went wrong (false positive). In a nutshell, the software development goal for the verifier consists in avoiding these situations altogether.

Given the mathematical and technical complexities of cryptographic voting protocols, developing a verifier directly from the protocol specification is a very big challenge. It requires advanced skills in both applied cryptography and software development. If unqualified personnel is in charge of this task, it is likely that the implemented test catalog will not form a complete verification chain, or that some tests are implemented incorrectly. In both cases, the conclusiveness of the verification report is weakened considerably.

To ensure the required functionality and software quality, we propose a two-step procedure for developing the verifier. The first step consists in deriving a specification document from the specification of the voting system. This task should be performed by cryptography experts that are familiar with voting protocols in general and with the specific technical details of the voting protocol at hand (possibly by the designers of the voting protocol). The main part of this document is the aforementioned test catalog, which together must form a complete verification chain. To assure the completeness of this chain, assembling the test catalog must be carried out with meticulous precision. To detect remaining gaps as early as possible, we also recommend applying a thorough reviewing process to this document. For maximal transparency, we also recommend the publication of this document.

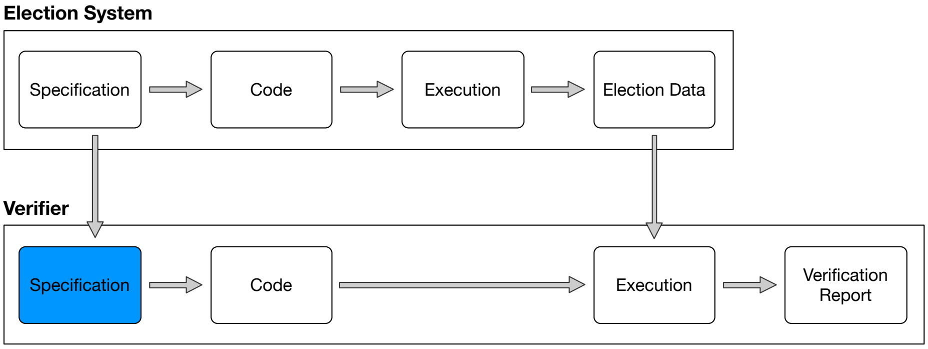


Fig. 6: Developing and executing the verifier based on a separate specification document.

Developing the actual software based on the verifier's specification is the second step of the proposed procedure (see Figure 6). This task can be delegated to a software engineer with only moderate background knowledge in cryptography and cryptographic protocols. To achieve general software quality properties, standard software design and coding principles should be applied to the development process. Code reviewing is another important method to establish the desired code quality. For maximal transparency, we also recommend to publish the source code and to invite the public to participate in reviewing the code.

Additional preconditions for developing the verifier are a precise interface description for obtaining the election data from the voting system and the availability of some meaningful test data. Both preconditions must be met by the developers of the voting system. Ideally, the test data also contains inconsistencies or flaws, such that the developed software can be tested for false positives and false negatives. Finally, it is also very important to implement a strict versioning policy, because even the slightest change in the voting system or in the election data may be enough to affect the proper functioning of the verifier.

3 Hybrid Election Processes

The election and verification processes as discussed so far are only directly applicable to the simple case of a purely electronic election with a single voting channel. The situation usually gets more complicated if multiple voting channels are offered simultaneously. The simplest way of handling multiple channels is to let the voters choose their preferred channel prior to an election. This leads to a decomposition of the electorate, which finally results in conducting multiple elections independently of each other. In this case, no channel coordination other than summing up the individual election results is necessary. If one of the channels is an electronic one, the verification can therefore be conducted in isolation using the process described in the previous section.

A more complicated situation arises if voters can choose the voting channel spontaneously during the election period. The composition of corresponding election processes is called a *hybrid election process*, and we will see in this section that extra precautions are necessary to handle this case properly. We are particularly interested in hybrid election processes because they correspond to the current plans in Switzerland of offering the electronic channel in addition to the two existing voting channels (postal mail, in person). The question that we want to address here is how to conduct the verification of the electronic votes, if postal voting or voting in person takes place simultaneously.

3.1 Extending the Election and Verification Processes

The major problem that arises in a hybrid election process is to ensure that no voter submits more than one vote over the available channels. This implies that using one channel for submitting a vote must disqualify the voter in every other channel. It is clear that implementing this seemingly simple principle requires accurate coordination between the channels. In practice, it turns out that the submission of multiple votes over multiple channels can not be avoided completely, even if doing so is illegal. If this happens, it should at least not be possible that two votes from the same voter are counted. Double votes from the same voter must therefore be eliminated—together with other invalid ballots—before starting the tallying process. This process, which is called *cleansing*, is a mandatory initial step of the post-election phase.

From the perspective of the election process model of Section 2.1, an additional input containing the list of disqualified voters is required to perform the cleansing of the submitted ballots before initiating the tally. This leads to the extended election process model of Figure 7. The actual electorate that is relevant for the tally is obtained from eliminating the disqualified voters from the electorate. The model depicted in Figure 7 also shows that the list of actual election participants is an additional output of the process. This list defines the disqualified voters in every other voting channel of the hybrid system. In the next subsection, we will see how to combine two or multiple such election processes into a hybrid election process.

The additional input and output documents in the extended election process model must be taken into account when performing the verification. Note that every single entry in each of these documents is highly critical, because they define somebody's right to submit a vote over some channel. Figure 8, which shows the extended verification process model, illustrates the inclusion of these documents. The purpose of the verification report is still the same, but since two additional inputs are now taken into account, a successful report also validates their contents.

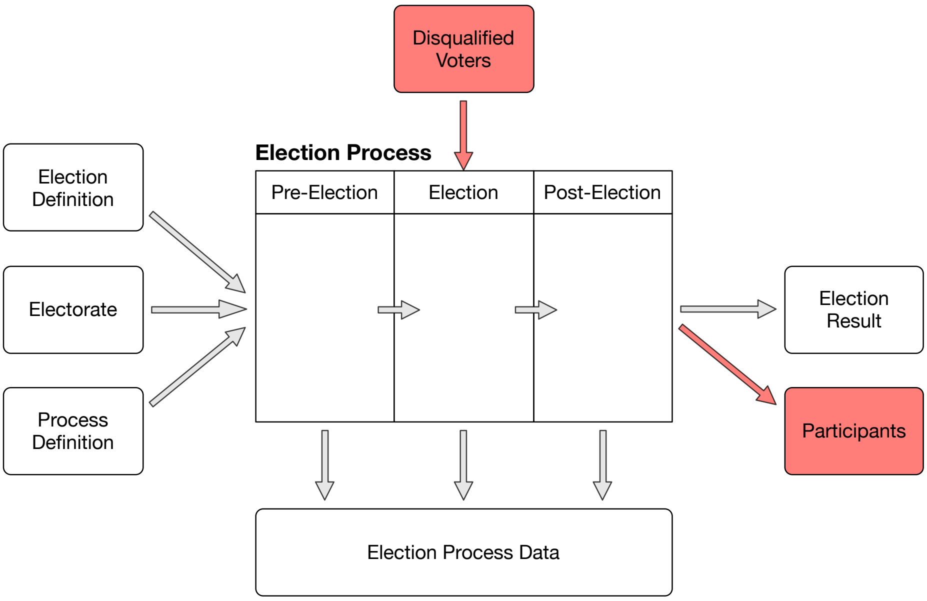


Fig. 7: Extended election process model for hybrid elections.

3.2 Composed Election Processes

Let's now have a closer look at actual compositions of multiple election processes. We will restrict ourselves to the simplest case of composing two alternative election processes. As we will see, the result of such a composition is again an election process, which can be further combined with other election processes. In this way, it is possible to construct recursive process models for more complicated combinations of three or more voting channels on the basis of the basic compositions described here.

For analyzing the composition of two election processes, we can distinguish two opposed cases. In the case of a *serial composition*, the temporal availability of the two channels is exclusive, i.e., the election period of the first election process strictly precedes the election period of the second process. Figure 9 depicts the hybrid process model obtained from a serial composition. It shows that the list of participants from the first channel defines the list of disqualified voters in the second channel. Note that the inverse data flow from the second channel back into the first channel is not required to guarantee the detection of double votes. Serial compositions are therefore relatively easy to handle properly.

More complicated situations arise in the case of a *parallel composition*, in which the election periods of the two processes overlap. In this case, the data exchange between the two channels is mutual. The resulting process model is depicted in Figure 10. It shows how the list of participants from each of the two channels is given as an additional input into the other channel. The problem here is that the same voter may appear in both lists, which must be taken into

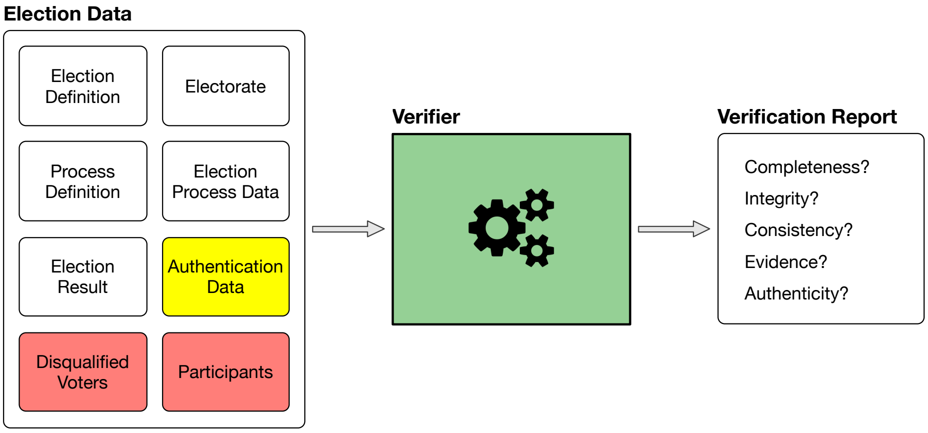


Fig. 8: Extended verification process model for hybrid elections.

account in each of the two cleansing processes. To handle such cases properly, there must be a clear policy of prioritizing one of the two submitted votes.

We see three different general strategies for defining such a policy. We will shortly discuss them in the remaining of this section. For this, we consider the use case from Switzerland, where an electronic voting channel is combined with a physical voting channel (postal mail). We assume that an electronic vote counts as "submitted" when the voter terminates the voting process, for example by clicking a button from the voting application's user interface. In case of submitting a paper ballot using postal mail, we assume that the vote counts as "submitted" when the ballot is registered at the polling station. Note that in Switzerland, submitting more than one vote is prohibited by law, regardless of the available voting channels. However, since voters are instructed to submit a paper vote in case of a problem encountered when submitting an electronic vote, enforcing this law will be difficult in practice. In other countries, for example in Norway, submitting multiple votes is explicitly allowed. In such a case, the last submitted vote overrides all previously submitted votes.

Prioritizing the Physical Channel. The rule here is as follows: if the same voter uses both channels to submit a vote, only the vote submitted over the physical channel will be counted. With this policy, paper votes can be counted regardless of the list of participants from the electronic channel. Therefore, the problem of eliminating double votes is only relevant for the electronic channel. Note that this situation is similar to a serial composition, in which the physical channel precedes the electronic channel. This policy is therefore relatively simple to implement. It is also compatible with a current practice in Switzerland, where administrative staff at the electoral office separates paper votes from the signed polling cards right upon receiving the paper ballot.

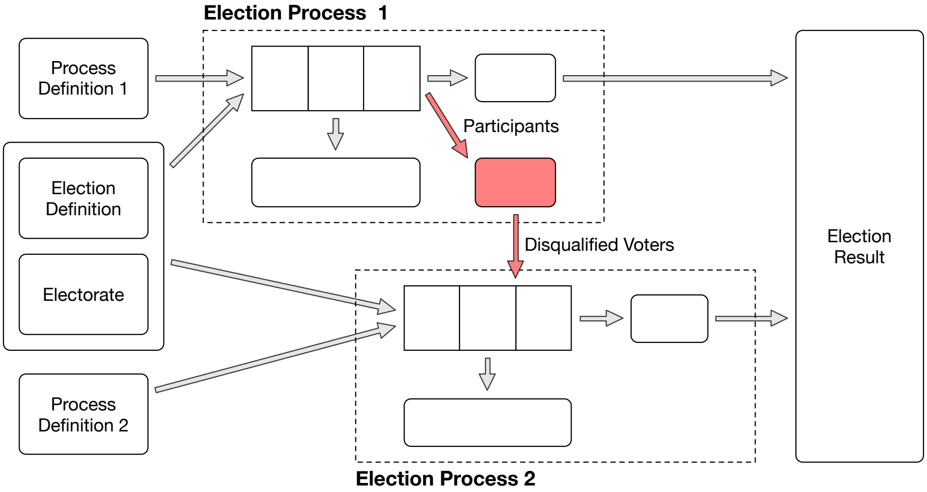


Fig. 9: Serial composition of two election processes.

Prioritizing the Electronic Channel. Here, the rule from above is applied in the opposite way, i.e., only the electronic vote of a voter using both channels is counted. The counting of the electronic votes can therefore be conducted regardless of the list of participants from the physical channel. This also simplifies the verification process, which can be conducted independently of the physical channel, but it makes the counting of the paper ballots at the polling station more complicated. For example, separating the paper votes from the signed polling cards must be postponed until the complete list of participants from the electronic channel is available.

Prioritizing the First or Last Submitted Vote. In this case, if someone submits two votes over both channels, only the first or the last submitted vote will be counted. This is the most complicated policy to implement, because the channels are mutually dependent on each other, i.e., exchanging both lists of participants according the Figure 10 is a mandatory precondition for eliminating double votes in both channels. The exchange of these lists can be done in two ways, either dynamically during the election phase or in a single step at the end of the election phase. In the dynamic case, the two voting channels may try to sort out double votes at the moment of receiving them, but a perfect synchronization is obviously very difficult to implement. Therefore, conducting the cleansing process at the end of the election phase is necessary in either case.

To enable the prioritization of either the first or the last submitted vote, timestamps must be added to the lists of participants, which define the exact moment of submitting the vote. The decision of keeping or ignoring a submitted vote is then based on these timestamps. Note the issuing reliable timestamps in an electronic context is a difficult problem on its own, especially if third parties must be able to verify the correctness of the timestamps in a conclusive way.

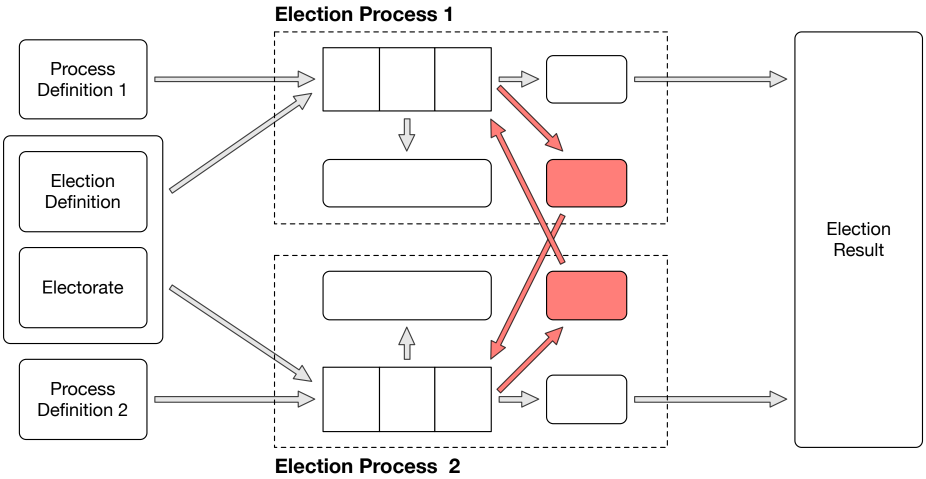


Fig. 10: Parallel composition of two election processes.

4 Conclusion

This paper is an attempt to define the universal verification process for electronic elections. The motivation for this paper comes from the observation that there is almost no practical experience with conducting actual verifications. On the other hand, since universal verifiability is commonly recognized as one of the most important counter-measures against all sorts of failures or attacks, almost everyone agrees that it must be implemented into future e-voting systems that are used for real political elections. Our analysis of the verification process in this paper shows that conducting an actual verification is more complex than it may appear at first sight. By discussing some of the most apparent questions and problems, we hope to provide some general technical guidelines for people in charge of implementing or organizing a verification process.

In most parts of the paper, for making our analysis and findings as widely applicable as possible, we have adopted a very general perspective. However, relative to a concrete voting system and application use case, many specific questions only arise if all the details about the cryptographic voting protocol, the technical system specification, and the political and legal contexts are available. Therefore, we can not answer these questions here, but we recommend not to underestimate the problems that may arise. More generally, we recommend to pay attention to the difficulties of the verification process well in advance. Election organizers should look at it as a separate important project, which also requires a careful planning, proper management, and adequate budget.

Acknowledgments. We thank the anonymous reviewers for their comments and suggestions. We are also grateful to Timo Bürk, Xavier Monnat, Jörg Schorr, Olivier Esseiva, and Anina Weber for helpful discussions, proofreading, and feedback. This research has been supported by the Post CH Ltd.

References

1. *Verordnung der Bundeskanzlei über die elektronische Stimmabgabe (VEleS).* Die Schweizerische Bundeskanzlei (BK), 2013.
2. B. Adida. Helios: Web-based open-audit voting. In P. Van Oorschot, editor, *SS'08, 17th USENIX Security Symposium*, pages 335–348, San Jose, USA, 2008.
3. D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. *IEEE Security & Privacy*, 6(3):40–46, 2008.
4. E. Dubuis, S. Fischli, R. Haenni, S. Hauser, R. E. Koenig, P. Locher, J. Ritter, and P. von Bergen. Verifizierbare Internet-Wahlen an Schweizer Hochschulen mit UniVote. In M. Horbach, editor, *INFORMATIK 2013, 43. Jahrestagung der Gesellschaft für Informatik*, LNI P-220, pages 767–788, Koblenz, Germany, 2013.
5. D. Galindo, S. Guasch, and J. Puiggalí. 2015 Neuchâtel's cast-as-intended verification mechanism. In R. Haenni, R. E. Koenig, and D. Wikström, editors, *VoteID'15, 5th International Conference on E-Voting and Identity*, LNCS 9269, pages 3–18, Bern, Switzerland, 2015.
6. I. S. Gebhardt Stenerud and C. Bull. When reality comes knocking – Norwegian experiences with verifiable electronic voting. In M. J. Kripp, M. Volkamer, and R. Grimm, editors, *EVOTE'12, 5th International Workshop on Electronic Voting*, number P-205 in Lecture Notes in Informatics, pages 21–33, Bregenz, Austria, 2012.
7. R. Haenni and R. E. Koenig. Universelle Verifizierung von Wahlen und Abstimmungen über das Internet. *SocietyByte*, June 2017.
8. R. Haenni, R. E. Koenig, P. Locher, and E. Dubuis. CHVote system specification. *IACR Cryptology ePrint Archive*, 2017/325, 2017.
9. K. Häni and Y. Denzer. Visualizing Geneva's next generation e-voting system. Bachelor thesis, Bern University of Applied Sciences, Biel, Switzerland, 2018.
10. G. Scalzi and J. Springer. VoteVerifier: Independent vote verifier for UniVote elections. Bachelor thesis, Bern University of Applied Sciences, Biel, Switzerland, 2013.
11. M. Volkamer, O. Spycher, and E. Dubuis. Measures to establish trust in Internet voting. In *ICEGOV'11, 5th International Conference on Theory and Practice of Electronic Governance*, Tallinn, Estonia, 2011.