E-voting: Individual verifiability of public boards made more achievable

Jérôme Dossogne Olivier Markowitch
Université Libre de Bruxelles
Fac. Sciences, Dept. Computer Sciences
Boulevard du Triomphe - CP212, 1050 Bruxelles, Belgium
jerome.dossogne@ulb.ac.be olivier.markowitch@ulb.ac.be

Abstract

In this paper we present two publishing methods for the votes and the result of an election, the *TreeCounting* and alternative *TreeCounting* method. These methods make the verifiability of public boards more achievable by publishing the result of an election as a tree. Both can be parametrized to increase the average number of times each node of that tree has been checked.

1 Introduction

In e-voting systems, any step of the chain of events from the setup of the system and the participation of individuals to the publication of the results is a potential place where fraud can occur. In this paper we propose to focus on the verifiability [1] process.

We define the individual verifiability property as the possibility for an individual alleged to participate in an election as a voter to check that his own vote is in the tally. While the purpose of this article is not to demonstrate how the user can find his own vote in the published list or even to design a voting protocol ensuring his right to stay anonymous, let us notice that such technique exists relying for example on designated verifier signatures [2, 4]. The universal verifiability [1] is the possibility for any individual alleged to participate in an election as a voter to check that every vote of that election was correctly recorded and that every recorded vote was taken into account in the global result of the election. The eligibility verifiability [1] property is the possibility for anyone to check that only eligible votes are included in the declared outcome. This paper focuses on the two first properties.

To achieve the individual verifiability property, most e-voting systems rely on the publication of the votes as a list on a public board [2, 3, 4, 5, 6]; the verifiability is achieved in two steps: (1) each voter checks that his vote is correctly published, (2) each voter sums all the published votes and checks whether the total is equal to the published one which means that his vote is taken into account and did influence the result.

In this paper we propose an alternative publishing method of the votes on a public board, we will refer to it as *TreeCounting*. In theory, when publishing the votes as a list, any participant could indeed perform the two steps mentioned above. In practice, when the number of participants exceeds a rather low threshold, we think that no human being can nor will perform the second step in an accurate manner. Of course, it could be possible to rely on tools to check the published sum, but the voters would have then to trust these tools that could have been programmed by others.

Therefore, we propose a method who will ensure that any individual can contribute to validating the sum of all the votes by calculating simple sums of a reasonable number of values.

This method does not completely solve the problem: a perfect method should allow any voter to validate the sum of all the votes without relying on anyone or anything but himself. However our method allows each participant to easily contribute, in a positive way, to this validation.

In short, we propose to publish the votes as a tree in which the leaves are labeled with the different votes and each internal node of the tree is labelled with the sum of the labels of its sons. Then each voter can choose to check one or several nodes (he checks whether the sum of all labels of the sons of a node a is equal to the label of that node a), if he agrees to the published values of the labels, he validates these values, and these validations appear on the public board.

Therefore, everyone and anyone can observe the coverage of the validations, how many times each node has been validated or not and thus can evaluate the trust he can have in the published results.

2 TreeCounting

2.1 Common listing method

Currently, most e-voting systems that use boards to publish the result of an election simply publish them as a list. For an election with l candidates or possible choices and n citizen, the publication can be modeled in the following form: n lines, each one representing a vote and an additional line with the result as illustrated with the table 1. In this example, for a ballot i, if a candidate m is the selected choice, then $c_{i,m} = 1$ and $\forall j \in [1, l], m \neq j : c_{i,j} = 0$. In this example, each voter can only vote for one candidate. The example can be generalized to any election where a voter can vote for n candidate since such election can be transformed into n elections where the voters can vote for only one candidate.

	Candidate 1	Candidate 2	...	Candidate l
Ballot 1	$c_{1,1} = 1$	$c_{1,2} = 0$	...	$c_{1,l} = 0$
Ballot 2	$c_{2,1} = 0$	$c_{2,2} = 0$	...	$c_{2,l} = 1$
...	...	...	...	...
Ballot n	$c_{n,1} = 0$	$c_{n,2} = 1$	...	$c_{n,l} = 0$
Result	$\sum_{i=1}^{n} c_{i,1}$	$\sum_{i=1}^{n} c_{i,2}$	...	$\sum_{i=1}^{n} c_{i,l}$

Table 1: Example of a simple listing method

To achieve individual verifiability, each voter has to perform the following two steps: (1) each voter checks that his vote is correctly published, (2) each voter sums all the published votes and checks whether the total for his candidate is equal to the published one. While the first step is relatively easy, the second one requires each voter to sum n numbers where n is the number of voters. Therefore, when the number of participant n exceeds a rather low threshold, we think that no human being has the ability to manually perform the second step.

The universal verifiability requires the second step to be applied on all the votes for all the candidates. Performing the eligibility verifiability requires to complete the voting scheme with additional feature since no one knows what the other votes are supposed to be. The second step requires the voter to sum $l * n$ digits where l is the number of candidates and n the number of voter. As we explained for the individual verifiability, while any voter is allowed to perform this verification, we think that no human being can nor will perform the second step.

In this paper, we propose a set of alternative publishing method that intends to help performing the second step.

2.2 The proposed method

2.2.1 Presentation

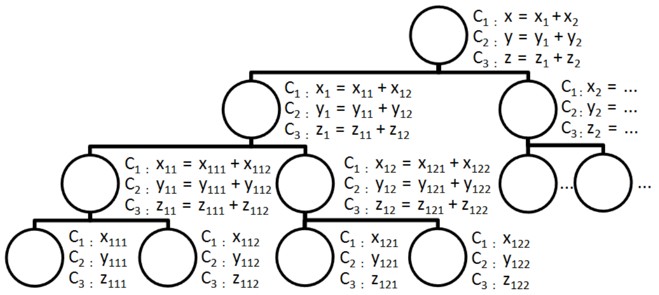

Figure 1: TreeCounting with a binary tree and 3 candidates

In the figure 1 we present another method to display the results. We define a vote as the expression of a choice by the voter. In our example, each voter has 3 choices $C_i, i \in [1,3]$, in other words, each voter can choose one candidate out of 3.

Instead of publishing the votes and the result as a list, we propose to publish the votes as a complete binary tree * in which the leaves are labeled with the different votes and each internal node of the tree is labeled with the sum of the labels of its sons as illustrated with the figure 1. We will refer to such trees as complete binary counting trees.

In our example, each node has the attributes x, y and z. x, y and z are used to count the number of votes respectively for the first, second and third candidate †. In a leaf, each of these attributes can either be equal to 1 or to 0. The results of the election is contained within x, y, and z of the root node of the tree.

In other word, the list previously published is now represented by the leaves of a tree and the published result is now displayed in the root of the tree. The internal nodes represent the intermediary calculations from the leaves to the root. All the information previously published still appear, we only add information, the result of the intermediary calculations.

From this point on, each voter can choose to check one or more nodes (he checks whether the sum of all labels of the sons of a node a is equal to the label of that node a), if he agrees to the published value of the label, he validates this value, and this validation appears on the public board.

This method can be generalized to any complete k-ary tree. In order to keep the calculations easily computable, k should be reasonably small.

2.2.2 Induced properties

Every property induced by the publication of the information via the common listing method [2, 3, 4, 5, 6] still exist since those information are still displayed and can be read/recovered as easily as in the listing.

Since we display more information, there is an overhead. Indeed, in a common listing method, only the votes are displayed. As we suggest to publish the vote as the leaves of a k-ary counting tree, the additional information is strictly inferior to the amount of information represented by the votes. For example, the worst case scenario

* *A binary tree in which every level, except possibly the deepest, is completely filled. At depth n, the height of the tree, all nodes must be as far left as possible.*

† In the figure, we numbered x y and z in each node in order to show the details of the calculations.

in term of overhead happens if we have a complete binary counting tree. In that case, the number of votes, thus of leaves, is $x = 2^n$ where n is the height of the tree. The total number of nodes in the tree including the leaves equals $\sum_{i=1}^{n-1} 2^i$ which is inferior to 2^n . In other word, even in the worst case scenario, the overhead in information is doubled. While this overhead might seems relatively excessive, its absolute size and cost are quite reasonable.

Here is a concrete example. With 10 billion votes [‡] in an election with 100 candidates, by encoding each number with 32 bits, the amount of information to store for the election is $10^9 * 100 * 32 = 800$ GigaBytes. As previously indicated, using a binary counting tree would increase this amount by less than the actual amount, thus the total amount of information would be inferior to 1.600 TeraBytes. A hard disk drive of such size can be bought in 2010 for less than 150€.

This method does not completely solve the problem: a perfect method should allow any voter to validate the sum of all the votes without relying on anyone or anything but himself. However our method allows each participant to easily contribute, in a positive way, to this validation.

Therefore, this method increases the universal verifiability in practice since anyone can perform the second step of the universal verifiability and thus can verify every vote was taken into account. Since the verification rate of the universal verifiability is increased, this method increases also the verification rate of the individual verifiability in practice. Indeed, since anyone can verify that all the votes were correctly taken into account, that same person can do the same verification for his own vote.

By adding a field in each node that indicates each time a node was validated by a voter, this method allows everyone to observe the coverage of the validations, how many times each node has been validated or not and thus to evaluate the trust he can have in the published results. Furthermore, a second field must be added to each node that indicates each time a node was considered non-valid.

Since voters are the one responsible of validating the tree, it is possible for a group of attackers to incorrectly validate the tree. However, each voter will validate only the internal nodes of the tree and has to identify himself to do so.

Since we wish to escape the psychological effect where each voter validates especially the nodes nearest to his vote, we suggest to publish the tree with colour based code relative to the score of each node (the score being the number of times it has been validated minus the number of times a voter indicated the node was invalid). This would help the voters selecting which node to check.

Obliviously, a complete k-ary counting tree can be created by using a complete k-ary tree instead of binary tree. Using such trees increases the coverage of the validation by reducing the number of internal nodes for a constant number of votes. Indeed the height of the tree and thus the amount of nodes decreases when k increases. Therefore, each voter agrees to check a constant number of nodes, the coverage of validation is increased. Of course, the complexity of validating one node increases also with k which could lead to an increased error rate.

2.3 Alternative method

2.3.1 Presentation

We illustrate this alternative method with the following example: an alternative *Tree-Counting* for an election of 3 candidates and 20 votes with 3 alternative binary counting trees. In our example, see figure 2, the first candidate has 8 votes, the second 5 and the third 7.

[‡]Note that the entire Earth's population is currently estimated by the United States Census Bureau to be around 6.8 billions

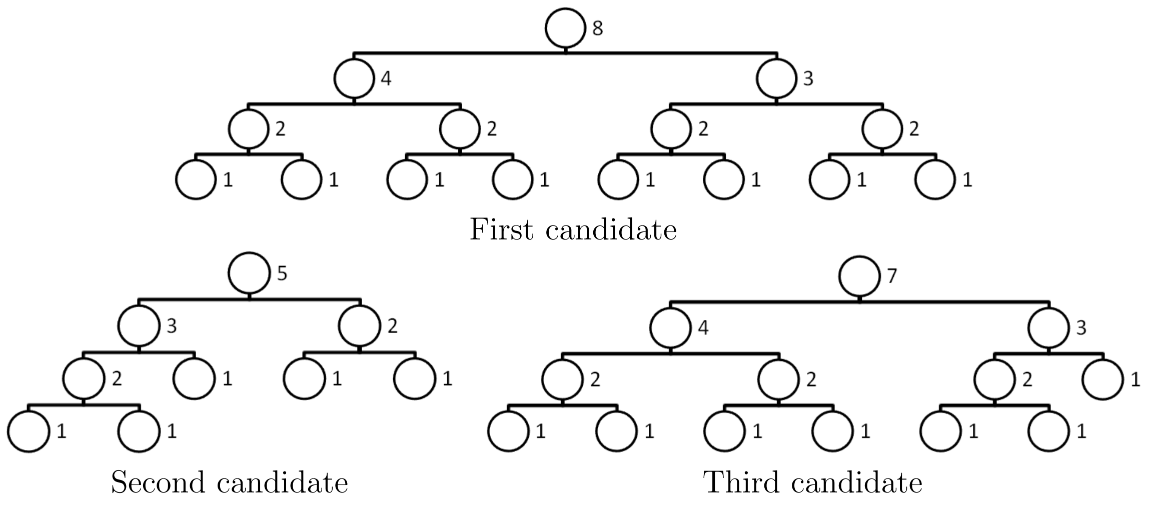

Figure 2: Alternative binary counting trees

First of all, as shown in our example, let us indicate that the alternative *TreeCounting* method can no longer be used when a voter has the possibility to vote for multiple candidates.

Indeed, instead of publishing the whole listing of votes for the l candidates as one k-ary complete tree, we publish that information as l k-ary complete trees and sort all the votes for each candidates in separate trees.

By doing so, the verification process speed is increased by tenfolds. The value in a node belonging to a perfect k-ary[§] subtree is predictable. If the node is the root of such a tree where the distance to the leaves is h, its value should be k^h. Knowing this, it is now very easy and fast for a voter to validate nodes, levels of a tree or even trees.

If predicting this value seems too complicated since it implies the use of power, tables of powers of k can be pre-distributed and pre-validated. With such table, a voter only has to compare the values in the tree with the values in his table. If the tree is not perfect, then the rightmost node of a level has to be validated summing the values in his sons since, as illustrated in our example, that node is not the root of a perfect k-ary subtree.

2.3.2 Induced properties

While the use of alternative complete k-ary counting trees allows to speed up the validation process, the alternative *TreeCounting* method can no longer be used when a voter has the possibility to vote for multiple candidates nor for elections where the voter has to indicate his order of preferences.

As we previously mentioned, the first case can be easily overcomed by transforming an election where a voter can vote for n candidates into n elections where the voters can vote for only one candidate. The second case is as easy to solve but more difficult to implement since each permutation has to be transformed into one possible choice. Therefore, in that scenario, if the number of candidates is too important, we do not recommend the use of *TreeCounting*.

We define the validation coverage of a tree as the average number of times each node of that tree has been checked. Therefore, the validation coverage of an election using *TreeCounting* is the average number of times each node of all counting trees of

[§] *Definition: A k-ary tree with all leaf nodes at same depth. All internal nodes have degree k.*

that election has been checked. Since each vote is present in only one tree, the alternative *TreeCounting* method could lead to an overall decrease of the election validation coverage. Indeed, in this method, if an individual only cares for his own vote and performs the individual verifiability process, the result would be an increase in the validation coverage only of the tree of his own candidate. In other word, while increasing the universal verifiability in practice is still possible with this method, the behavior of the individuals could lead to a diminished election validation coverage. This could be avoided by combining the *TreeCounting* and alternative *TreeCounting* method. If we publish the counting tree and the alternative counting trees we provide the voter with the right of double checking the implication of his vote.

If the implemented system forbids a voter to validate only the label of one candidate in a node instead of all the labels of that node, notice that the *TreeCounting* method should improves more the universal verifiability process than the alternative *TreeCounting* since we suppose that voters will be more inclined to validate only the tree of their candidate in the latter.

3 Conclusion

In this paper, we present two publishing methods for the votes and the result of an election, the *TreeCounting* and alternative *TreeCounting* method. Both can be parametrized to increase the average number of times each node of that tree has been checked, i.e. the validation coverage. The two methods can be easily used at the same time leading to a more flexible system. Such system would benefits from a more practical verifiability process thanks to the *TreeCounting* and from a more practical and faster verifiability process thanks to the alternative *TreeCounting*.

From the observation that publishing the votes as a simple list leads only to a theoretical individual of universal verifiability, we think that implementing the proposed methods and thus increasing the verifiability in practice is worth the effort.

References

[1] Marc Ryan, Analysis of Verifiability in Electronic Voting, VOTE-ID., 2009

[2] Jérôme Dossogne and Olivier Markowitch, Voting With a Tripartite Designated Verifier Scheme Based On Threshold RSA Signatures, Proceedings of the 30th Symposium on Information Theory in the Benelux (WIC09), vol. 1, 113-118, 2009

[3] Sung-Hyun Yun and Hyung-Woo Lee, The Large Scale Electronic Voting Scheme Based on Undeniable Multi-signature Scheme, Computational Science and Its Applications ICCSA 2005. Lecture Notes in Computer Science, vol. 3481/2005, 391-400, 2005

[4] Olivier Markowitch and Emmanuel Dall'Olio, Voting with designated verifier signature-like protocol, IADIS International Conference WWW/Internet 2004, 295-301, 2004

[5] Wei-chi Ku and Chun-ming Ho, An e-Voting Scheme with Improved Resistance to Bribe and Coercion, Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04), 113-116, 2004

[6] Indrakshi Ray and Indrakshi Ray and Natarajan Narasimhamurthi, An Anonymous Electronic Voting Protocol for Voting Over The Internet, Proceedings of the Third International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS '01), 21-22, 2001