Efficiency Comparison of Various Approaches in E-Voting Protocols

Oksana Kulyk[1] and Melanie Volkamer[1,2]

[1] Technische Universität Darmstadt/CASED, Darmstadt, Germany
name.surname@secuso.org
[2] Karlstad University, Karlstad, Sweden

Abstract. In order to ensure the security of remote Internet voting, the systems that are currently proposed make use of complex cryptographic techniques. Since these techniques are often computationally extensive, efficiency becomes an issue. Identifying the most efficient Internet voting system is a non-trivial task – in particular for someone who does not have a sufficient knowledge on the systems that currently exist, and on the cryptographic components that constitute those systems. Aside from these components, the efficiency of Internet voting also depends on various parameters, such as expected number of participating voters and ballot complexity. In this paper we propose a tool for evaluating the efficiency of different approaches for an input scenario, that could be of use to election organizers deciding how to implement the voting system.

1 Introduction

Both vote secrecy and verifiability of the result are the crucial requirements in Internet voting. For ensuring them, various cryptographic techniques have been proposed. The two common approaches to anonymise the votes are verifiable mix net and homomorphic tallying. Both these approaches, being versatile in use have been widely employed in the literature, and also implemented in systems used in practice [2, 7, 8]. Moreover, these approaches are interchangeable in some of the voting schemes: the Helios system, which used both mix net and homomorphic tallying approaches in its versions, is an example of such interchangeability. Thus, the choice of either mix net or homomorphic tallying approach does not have an impact on the security of the scheme. In such cases, the decision to use either one of them for ensuring vote secrecy has to be made by the election organizers. One of the important criteria is the efficiency of the resulting scheme.

In this paper we implement a prototype tool, that enables comparing the efficiency of both these approaches by estimating the theoretic performance of corresponding calculations. This tool is then supposed to support the election organizers to appropriately implement the voting system, by choosing the most efficient anonymisation approach.

We evaluate the efficiency of the anonymisation using different types of ballots: namely, different kinds of approval voting, divisive (weighted) voting and

ranked voting. Each one of this type can be used either with the mix net approach, or homomorphic tallying with different kinds of validity proofs. We count the operations that require most performance and implement a prototype tool[3] that, using the formulas we provide, enables estimating the efficiency of different approaches given a specific election setting, thus helping to decide which of these approaches would be the most effective in this setting.

The paper is structured as follows. We outline the methodology that we use for estimating the efficiency in Section 2. We provide details on individual anonymisation approaches in Section 3. Finally, we describe the prototype tool we developed for evaluation in Section 4, and present the evaluation results of various election settings in Section 5.

2 Methodology

In this section we describe the methodology we used in order to estimate the time needed for the computations. First, we identify the appropriate election phases, which efficiency can differ depending on the anonymisation approach that is used. Then we describe the way to estimate the time for the computations during those phases.

2.1 Election phases

The following election stages necessarily differ, depending on the anonymisation approach that is used:

Voting The voter uses her private device in order to prepare and cast a vote over the internet. Depending on the anonymisation method, different proofs of well-formedness need to be computed, in order to prevent casting invalid ballots.

Validation The voting system verifies the validity of each cast vote, by verifying the vote validity proofs that are appended to votes during the vote casting. The system can start validating the votes directly after they are being cast, and it needs to be fully completed before the tallying can start.

Tallying After all the votes have been cast and verified, the result is being tallied. This includes performing the mixing in the case of mix net based approach, and finding the discrete logarithm in the case of homomorphic tallying. After the anonymisation, the results are being decrypted. The number of ciphertexts to be decrypted also depends on the type of the anonymisation. We assume, that both the tasks of mixing and of decrypting are performed by the same set of trustees.

[3] The tool will be made open-source following the publication.

2.2 Time estimations

Let $\mathbb{G}_q$ be a cyclic group with order q, that is used for all the calculations in the election. In order to estimate the efficiency of different anonymisation approaches, we rely on the efficiency of performing one exponentiation in $\mathbb{G}_q$ on a computer that runs the election. This value depends on several factors: not just on the processor used, but also on the security parameters of the election, such as the bit size of underlying group elements and exponent order q, or on whether $\mathbb{G}_q$ is a subgroup of $\mathbb{Z}_p$ for a prime p, or an elliptical curve. Therefore, ideally it should be an input from the election organizers for each individual case.

This value, further denoted as $RExp$ is then used as a basis for estimating the efficiency of individual operations. For further optimising the estimations, we also consider the special kinds of exponentiations that might speed up the calculations, as well as the possibilities to precompute some of the values in advance.

Calculation optimizations Aside from calculating the exponentiations in a straight-forward way, one can apply other algorithms developed for calculating special kinds of exponentiations. These algorithms, outlined in [16], can perform computations in a more efficient way than computing each exponentiation separately. The special kinds of exponentiations that are relevant for this work are as follows:

Fixed-base exponentiations (FBExp): Computing multiple exponentiations $g^{e_1}, ..., g^{e_m}$ for a single base g.

Multi-exponentiations (MExp(m)): Computing the product of exponentiations $\prod_{i=1}^{m} g_i^{e_i}$.

In our calculations we assume, that the voting system can rely on precomputations, while the voting client does not. We further assume the exponent size of 256 bits, which is the recommended size for both integers and elliptical curves according to `keylength.org`. We then use following heuristics to determine the type of exponentiations used in calculations for optimal efficiency:

- the voting client uses multiexponentiations where available,
- the voting system uses fixed-base exponentiations where available,
- for multiexponentiations with large values of m, the product $\prod_{i=1}^{m} g_i^{e_i}$ is calculated in smaller batches[4] as $p_1 \cdot ... \cdot p_{\lceil m/7 \rceil}$ with $p_i = \prod_{j=7(i-1)+1}^{7i} g_j^{e_j}$.
- for fixed-base exponentiations, the representation of an exponent $e = \sum_{i=0}^{t-1} e_i \cdot b^i$ is used, with $b = 16$, $t = 64$, $e_i < b \forall i = 0, ..., t-1$.

The time needed for both multiexponentiation and fixed-base exponentiation is then determined relatively to $RExp$, with $MExp(2) = 1.16 \cdot RExp$, $MExp(7) = 1.64 \cdot RExp$, $FBExp = 0.19 \cdot RExp$.

[4] We consider splitting the product in batches of size seven, due to its optimal performance.

Considerations about pre-computations A decision can be made by the election organizers, to perform some of the needed computations in advance, thus speeding up the computations during the election. As we assume, that no pre-computations can be done by the voter, the operations that can be pre-computed are as follows.

Special-kind of exponentiations As already mentioned, special algorithms can be employed for performing some parts of calculations more efficiently. In particular, they can be of use when having to calculate a large number of exponentiations with common base, thus speeding up each new exponentiation with this base significantly.

Discrete logarithm As in the homomorphic tallying approach, the calculation of a discrete logarithm is necessary given a set of values $g_1^{x_1}, ..., g_C^{x_C}$ with C as a total amount of resulting ciphertexts. Thus, for each g_i one could use a precomputed table of values (x, g_i^x) for all possible values of x.

Mix net matrix commitments Given the mix net scheme in [20], a substantial part of the computations can be performed without the knowledge of ciphertexts that are about to be shuffled. We therefore assume, that the voting system performs precomputations that would allow to shuffle the votes from all eligible voters.

Parallelisation The operations performed by a single entity that we consider can be parallelised, by distributing the calculations into different parts and combining the result. This is especially trivial for homomorphic tallying approaches, where the tasks of verifying the individual validity proofs or finding the discrete logarithm results can be easily distributed. For the mix net approach, the operations that are needed for either calculating a proof of shuffle or for verifying it can be parallelised as well with an appropriate implementation. For the sake of simplicity, we consider that either all of the operations are parallelised using the same number of processors, or none is.

3 Individual calculations

In this section we provide the formulas that determine the estimated time needed for calculation of decryption of the final result, as well of specific anonymisation approaches.

3.1 Mix net

One of the anonymization methods considered is a mix net scheme, whereby the input list is being shuffled by each one of the trustees in turn, so that the correspondences between the ciphertexts in the input and output lists are hidden. Each of the ciphertexts in the output list is being decrypted and added to the final tally according to the ballot rules.

As long as at least two nodes keep the correspondences between the shuffled lists secret, it is unfeasible to connect any ciphertext in the original list to its correspondence in the final resulting list. In order to provide robustness against faulty mix nodes, a reencryption mix net scheme is used, and for ensuring that the ciphertexts are shuffled correctly and not replaced by manipulated votes, the proof of shuffle is attached by each mix node. We chose to include the proof of shuffle suggested by [20, 22] due to it being to our knowledge the most efficient algorithm, the implementation and detailed specification of which is available for open usage [23]. For the mix net scheme, the efficiency of calculating the proof of shuffle for C ciphertexts in terms of exponentiations is $(C+2)RExp + 2C \cdot MExp(2) + MExp(C+1)$ for the offline phase (i.e. that can be precomputed), and $3MExp(C+1)+2C \cdot FBExp$ for the online phase. The efficiency of verifying such proof is $MExp(C)+RExp+MExp(C+2)+C \cdot MExp(3)$ exponentiations for the offline phase, and $MExp(C) + 3MExp(C+2)$ for the online phase.

Note, that using the mix net based approach for anonymizing the votes does not place any restriction on the ballot type that is used; further, as long as individual vote can be encrypted in a single ElGamal ciphertext[5], the efficiency of the anonymization does not depend on ballot complexity.

3.2 Homomorphic tallying

The second approach is to avoid decrypting individual votes, while aggregating them instead, and decrypting only the aggregated result. This is possible if homomorphic cryptosystem is used to encrypt the votes, which is usually exponential ElGamal. It follows, that the homomorphic tallying approach is suitable, whenever the final tallying result can be represented as the sum of individual votes. Furthermore, additional zero-knowledge proofs have to be implemented, that allow to check for vote validity upon vote casting prior to aggregating the votes, in order to exclude the possibility of overvoting or negative voting. Therefore, in this section we consider ways to prove the validity of votes cast according to different ballot types.

Let N be a number of voters, $C_1, ..., C_L$ available candidates. For each ballot type, we consider the valid representation of a single vote, and possible values of the election result. The first value is crucial in proving the validity of a single vote. The second is useful in calculating the final result: that is, more possible combinations of votes would mean that more calculations have to be made for calculating the discrete logarithm.

Given $v_1, ..., v_L$ as the number of votes given for each candidate by a single voter, there are various approaches to encode and encrypt this choice. As such, the proofs of validity suggested in [9], encode the votes such that a single ciphertext results for each voter. Proofs by Joachim [13] and the proofs used in Helios voting system [1,11], namely the v4 version, result in L ciphertext, whereby votes for each candidate are encoded separately; while the ciphertexts in Helios are encoded as g^{v_i} for the same generator g, proofs in [13] encode the votes as $g_i^{v_i}$ for

[5] We consider it to be realistic in most cases

different generators. The number of ciphertexts is important for the efficiency of decryption and computations of the discrete logarithm at tallying.

Also note, that some of the methods proposed make use of a verifiable mix net scheme. Thus, we denote the time needed to prove the validity of shuffling C ciphertexts as $MixProve(C)$, and the time needed to verify such proof as $MixVerify(C)$. In our calculations we assume, due to considerations outlined earlier, that the scheme in [22] is used. However, the calculations are slightly different: first, there is no offline phase; second, due to the fact that the voting system has to verify a large amount of shuffles of the same ciphertexts. The resulting functions are $MixProve(C) = 4 \cdot MExp(C+1) + 2N \cdot FBExp + (C+2) \cdot RExp + 2C \cdot MExp(2)$, $MixVerify(C) = 2 \cdot MExp(C) + 3 \cdot RExp + MExp(C+1) + C \cdot MExp(2) + (4C+6) \cdot FBExp + RExp + 2 \cdot MExp(C+2)$.

Approval voting $k_{min}...k_{max}$ of L The most commonly used type of ballots can be grouped together as *Approval Voting*, whereby the voter is allowed to select at least k_{min}, at most k_{max} candidates. Thus, the single vote is conforming to the election rules, if it is of the form $\{v_1, ..., v_L : v_i \in \{0,1\}, \sum_{i=1}^{L} v_i \in [k_{min}, k_{max}]\}$; and the set of all possible election results is $\{v_1, ..., v_L : v_i \in [0, N], \sum_{i=1}^{L} v_i \in [N \cdot k_{min}, N \cdot k_{max}]\}$. Common elections that fall under this type are "Yes/No" elections (with $L = 1$, $k_{min} = 0$, $k_{max} = 1$, or 1 of L elections with $k_{min} = k_{max} = 1$. In Tables 1 to 3 we summarize the proofs of validity of such ballots that exist in the literature, together with the number of resulting ciphertexts. Note, that together with proofs for the general case $k_{min}...k_{max}$ of L, a number of proofs tailored to special cases, such as $k_{min} = k_{max} = k$, or $k_{min} = 0$, $k_{max} = L$, has been developed.

Table 1. Homomorphic tallying approaches, Approval Voting: Proof efficiency

Schema	Parameters	Proof
[11]	$k_{min}...k_{max}$ of L	$(2L+2) \cdot RExp + 2(k_{max} - k_{min} + 1 + L) \cdot MExp(2)$
[11]	$0...L$ of L	$2L \cdot MExp(2) + 2L \cdot RExp$
[13]	$k_{min}...k_{max}$ of L	$L \cdot RExp + MixProve(L + k_{max}) + L \cdot MExp(2) + MExp(L+1)$
[13]	k of L	$L \cdot RExp + MixProve(L+k) + L \cdot MExp(2) + MExp(L+1)$
[13]	$0...k_{max}$ of L	$L \cdot RExp + MixProve(L + k_{max}) + L \cdot MExp(2)$
[9]	k of L	$2 \cdot MExp(3k+2) + 2RExp + 1 + MExp(2)$
[9]	$0...L$ of L	$3RExp + MExp(L+2) + MExp(2)$

Divisive voting (t, T) of L The voter is allowed to distribute a total of T votes to L candidates, whereby each candidate can get up to t votes. This kind of elections is particularly relevant for shareholders elections, whereby each voter $i = 1, ..., N$ has a total of T_i votes to distribute, with T_i representing the amount of possessed shares. Without loss of generality, assume that $T_i = T \; \forall i = 1, ..., N$

Table 2. Homomorphic tallying approaches, Approval Voting: Verification efficiency

Schema	Parameters	Verification
[11]	$k_{min}...k_{max}$ of L	$(4L + 2 + 2k_{max} - 2k_{min}) \cdot (FBExp + RExp)$
[11]	$0...L$ of L	$4L(FBExp + RExp)$
[13]	$k_{min}...k_{max}$ of L	$MixVerify(L + k_{max}) + (4L + 4 + 2k_{max} - 2k_{min}) \cdot FBExp + (2L + 4 + 2k_{max} - 2k_{min}) \cdot RExp$
[13]	k of L	$MixVerify(L + k) + (4L + 2) \cdot FBExp + (2L + 2) \cdot RExp$
[13]	$0...k_{max}$ of L	$MixVerify(L + k_{max}) + 3L \cdot FBExp + 2L \cdot RExp$
[9]	k of L	$(3k + 5) \cdot FBExp + 3 \cdot RExp$
[9]	$0...L$ of L	$(L + 5) \cdot FBExp + 3 \cdot RExp$

Table 3. Homomorphic tallying approaches, Approval Voting: Ciphertexts for decryption and fixed-base precomputations

Schema	Parameters	Number of ciphertexts	Fixed-base precomputations
[11]	$k_{min}...k_{max}$ of L	L	128
[13]	$k_{min}...k_{max}$ of L	L	$64 \cdot (2L + k_{max} + 2)$
[9]	k of L	1	$128 + 64 \cdot (3k + 3)$
[9]	$0...L$ of L	1	$128 + 64 \cdot (3 + L)$

is the same for all voters. A variant of this type of ballot $(t, 0...T)$ of L allows not to distribute all the T votes.

As such, according to the election rules, a single vote must lie in the set of $\{v_1, ..., v_L : v_i \in [0, t], \sum_{i=1}^{L} v_i = T\}$. The set of all the possible election results can be defined as $\{v_1, ..., v_L : v_i \in [0, Nt], \sum_{i=1}^{L} v_i = TN\}$.

The proof by Groth et al. supports only the variant of $t = T$. For $t \leq T$, a proof of validity was developed by Joachim et al. In the Helios implementation, it is only possible to conduct elections with $T = L \cdot t$, although supporting elections with $T < L \cdot t$ is possible with additional modifications[6]. The efficiency of individual proofs is summarized in Tables 4 to 6.

Table 4. Homomorphic tallying approaches, Divisive Voting: Proof efficiency

Schema	Parameters	Proof
[11]	(t, T) of L	$2Lt \cdot MExp(2) + (2L + 2) \cdot RExp$
[11]	$(t, 0...T)$ of L	$(2Lt + 2T) \cdot MExp(2) + (2L + 2) \cdot RExp$
[11]	$(t, 0...Lt)$ of L	$2Lt \cdot MExp(2) + 2L \cdot RExp$
[13]	$(t, 0...T)$ of L	$L \cdot RExp + MixProve(Lt + T) + L \cdot MExp(2)$
[9]	(T, T) of L	$MExp(L + 1) + MExp(5L + 1) + RExp + MExp(2)$

[6] Such modifications would require computing and verifying additional zero-knowledge proofs for all questions of the election, in order to verify, that the sum of all votes of the election does not exceed T

Table 5. Homomorphic tallying approaches, Divisive Voting: Verification efficiency

Schema	Parameters	Verification
[11]	(t, T) of L	$2 \cdot (FBExp + RExp) \cdot (Lt + L + 1)$
[11]	$(t, 0...T)$ of L	$2 \cdot (FBExp + RExp) \cdot (Lt + L + T + 1)$
[11]	$(t, 0...Lt)$ of L	$2L(t + 1)(FBExp + RExp)$
[13]	$(t, 0...T)$ of L	$MixVerify(Lt + T) + 3L \cdot FBExp + 2L \cdot RExp$
[9]	(T, T) of L	$(4 + 5L) \cdot FBExp + 3 \cdot RExp$

Table 6. Homomorphic tallying approaches, Divisive Voting: Ciphertexts for decryption and fixed-base precomputations

Schema	Parameters	Number of ciphertexts	Fixed-base precomputations
[11]	(t, T) of L	L	128
[13]	$(t, 0...T)$ of L	L	$64(Lt + 2L + T + 3)$
[9]	(T, T) of L	1	$64(2 + 5L)$

Ranking k of L (Borda) In this ballot type, the voter is to assign the ranks 1 to k to k out of L candidates. The ranks from voters are summed up for each candidate to determine the election result.

Groths method offers only a solution for $k = L$. The proofs used in Helios system cannot guarantee the validity of the ballot: while one is able to proof that each individual vote lies in R, and the sum of all given votes equals $\sum_{i \in R} i$, in current implementation there is no way to guarantee that each candidates gets a unique rank.

A set of valid single votes therefore is defined as $\{(v_1, ..., v_L) : v_i \in 0 \cup R; \{v_i : v_i \neq 0\} = R\}$. The set of all possible election results is then $\{(v_1, ..., v_L) : v_i = \sum_{j=1}^{N} x_{ij}, x_{ij} \in 0 \cup R \; \forall x \in R : |(i, j) : x_{ij} = x| = N\}$. The efficiency of individual proofs is summarized in Tables 7 to 9

Table 7. Homomorphic tallying approaches, Ranking Voting: Proof efficiency

Schema	Parameters	Proof
[13]	k of L	$k \cdot MixProve(L+1) + MixProve(L) + (2L+1) \cdot RExp + MExp(L + 1)$
[9]	L of L	$2 \cdot RExp + MExp(L + 1) + MExp(2)$

3.3 Distributed decryption

Regardless of the anonymisation approach that is used, the vote secrecy also heavily relies on the decryption process, that ensures that only the anonymised ciphertexts are being decrypted. For this purpose, verifiable distributed threshold secret sharing is employed, that enables decryption only if a threshold amount of trustees collaborate, while ensuring that no single entity is in posession of

Table 8. Homomorphic tallying approaches, Ranking Voting: Verification efficency

Schema	Parameters	Verification
[13]	k of L	$k \cdot MixVerify(L+1) + MixVerify(L) + (3L+2) \cdot FBExp + (2L+2) \cdot RExp$
[9]	L of L	$(4+L) \cdot FBExp + 3 \cdot RExp$

Table 9. Homomorphic tallying approaches, Divisive Voting: Ciphertexts for decryption and fixed-base precomputations

Schema	Parameters	Number of ciphertexts	Fixed-base precomputations
[13]	k of L	$L(k+1)$	$64(2k+L+4)$
[9]	L of L	1	$64(2+L)$

a secret key. A commonly used method is the threshold distributed ElGamal key generation followed by distributed verifiable decryption, as described in [18]. Depending on the anonymisation method in use, the number of ciphertexts to be decrypted varies, together with the efficiency of the decryption. For small number of ciphertexts ($C < 50$), the efficiency of the decryption phase can be estimated as $(C(t_r - 1) + 1) \cdot FBModExp + C(t_r - 1) \cdot MExp(2) + (2Ct_r + 1) \cdot RModExp$, requiring the precomputation of 64 exponentiations; for larger amounts of ciphertext, however, the optimal estimation would be $(2C(t_r - 1) + 1) \cdot FBModExp + C(t_r - 1) \cdot MExp(2) + (Ct_r + C + 1) \cdot RModExp$ with the precomputation of $64 \cdot (1 + t_r)$ exponentiations.

4 Prototype evaluation tool

In this section we describe the tool implemented for the efficiency evaluation, based upon the input of previous sections, and provide the efficiency evaluation of various election settings.

4.1 Relevant parameters

We take a look at different parameters that influence the efficiency of the electronic voting, based upon the formulas we derived in Section 3. Depending on the anonymisation approach that is used, different kinds of parameters may or may not play a role in how long do the different stages of the election take.

Number of voters As one could expect, the main parameter that determines the efficiency of the election scheme, both for the mix net and homomorphic tallying approaches, is the number of voters that participate in the election. For the evaluation of precomputations that need to be done before the elections, the upper bound of participating voters is needed. For this, a total amount of eligible voters can be taken. For the efficiency estimation of the validation and tallying

phases, the actual amount of participated voters is required, an expected value of which can be based i.e. on previous voter turnout.

In presence of multiple voting districts participating in a single election, several alternatives exist on how to implement the system. The first alternative would be to run the election in a centralized way, whereby all the votes are being stored, processed and tallied by a single central server, while the second way would be for the each voting district to run a separate instance of the voting system themselves. Depending on the chosen approach, one could estimate either the performance of centralized system or of a single district, by inputting the parameters for the corresponding voting system instance.

Number of trustees This parameter is most important for evaluating the efficiency of a mix net based approach, since the trustees have to act as mix nodes. Furthermore, number of trustees also has an effect on the efficiency of distributed decryption of the result. Given the assumption that more than half of T trustees have to be honest, we set the threshold value as $t_r = \lfloor T/2 \rfloor + 1$. In case of mix net based approach, given the fact that at least one honest mix node has to participate, we set the number of mix nodes as $t_m = T - t_r + 1$

Number of candidates and other ballot-specific parameters The number of candidates or options is relevant for evaluating the efficiency of homomorphic tallying approach. It has an effect on the efficiency of proof of ballot validity, as well as on the total amount of possible election result - that is, on the complexity of calculating the discrete logarithm of the final result. Furthermore, number of candidates also influences the number of ciphertexts to be decrypted in some of the homomorphic tallying approaches. The same considerations hold for other ballot-specific parameters, as outlined in Section 3.2.

4.2 Software

A tool for comparing the schemes described above depending on input of the election parameters was implemented, using Java language. For the calculations we used the formulas for homomorphic tallying approaches mentioned in Section 3.2, mix net scheme from [20,22], as mentioned in Section 3.1, and verifiable decryption scheme mentioned in Section 3.3. Upon entering the input (see Figure 1), the tool computes the execution time needed for each one of the available schemes, as explained in Section 2. In our example calculations we consider the duration of $3ms$ for a single exponentiation, which roughly corresponds to the performance of a Macbook Pro Laptop using multiplicative group $\mathbb{G}_q \subset \mathbb{Z}_p$ of order q with p, q primes with bit lengths of 2048 and 256 respectively.

5 Evaluation of example settings

In this section we demonstrate the workings on the tool by selecting appropriate examples of the election settings, and showing how the efficiency of various approaches for this settings is estimated.

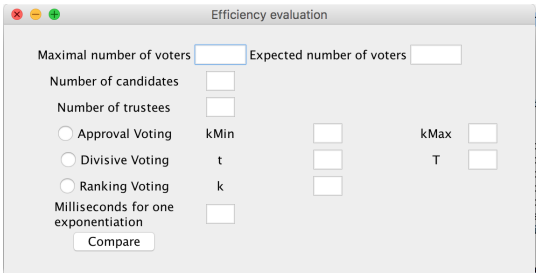

Fig. 1. Evaluation tool prototype interface

5.1 Description of example settings

We provide an example for the evaluation of an election setting, using the ballot types described in Section 3. Some of these settings are based on the public data from past elections[7], that were conducted using Internet voting. Others are are examples constructed by us for this evaluation specifically, which, however, might also be relevant for the elections, conducted in practice.

Approval voting: Estonian elections During the 2011 parliamentary elections, a total of $140,846$ out of the registered $913,346$ voters chose to vote via electronic means. A total of 789 candidates registered, out of which the voters were supposed to make their choice [4, 5]. The decryption key was distributed between 7 trustees. Thus, we evaluate an approval vote with "1 of 789" ballot.

Approval voting: Norway elections We consider the local elections in Norway in 2011 [14]. As the actual election rules are rather complex, in our analysis we consider the distribution of seats between the parties, without paying attention to personalised votes. In 2011, a total of $167,506$ out of $27,738$ eligible voters have cast their vote electronically, and 21 parties participated. There were a total of 10 trustees.

Approval voting: IACR elections The International Association for Cryptologic Research uses electronic voting for their internal elections. We consider the election of 2012 [12], where the voters had to cast their vote for any number of candidates out of registered 5. A total of 518 voters participated (out of 1530 eligible), and there were 3 trustees.

Approval voting: Boardroom voting A special kind of election setting involving small groups of voters, often referred to as boardroom voting, is the one

[7] Note, that the parameters in our examples may not correspond precisely to the real data.

where the roles of trustees are taken over by the voters themselves [10, 15]. For the evaluation of this setting, we chose the parameters of 30 participating voters, and, correspondingly, 30 trustee, voting on a 1 out of 5 ballot.

Approval voting: Swiss elections Switzerland has been conducting e-voting elections and referendums in some of its cantons for many years. As an example, we consider the data from one of the referendums, given votes cast using the Geneva voting system. In 2015, a total of 14052 votes were cast electronically using this system, out of eligible 119252 voters [19]. We assume 4 as the number of trustees, given the 2011 report [17].

Divisive voting As we are not aware of any real-world e-voting election that uses the divisive voting method, we had to construct an example by ourselves, partially using the data from traditional voting elections. Namely, we base our example election setting on the local Hesse elections [21], whereby the voters had to distribute 71 vote to 502 candidates, giving at most 3 votes to each candidate, and a total of 44385 out of 101666 eligible voters participated. We assume the participation of 3 trustees.

Ranking voting Similarly to the divisive voting ballot, we were unable to find data from a real-world election with this type of ballot. The closest example would be the elections in Australia, that also use the ranking voting ballot for their election, albeit using a different tallying method as opposed to Borda voting. We therefore use the parameters similar to the Victorian state elections [3,6] for our example: a total of 1121 participating voters, 7 trustees, and 40 candidates to be ranked.

5.2 Results and discussion

The evaluation results for all the settings, showing the estimated performance for expected number of voters, are given in Table 10. For setting and each election stage, the approach most efficient during this stage is marked in bold.

As one can see from it, in many of the cases, with the exception of simple ballots like "yes/no" elections, or approval voting elections with relatively small number of available options, the mixnet approach outperforms all the approaches based on homomorphic tallying. This can be explained by the fact, that the efficiency of this approach does not depend on the ballot complexity. The large number of trustees, however, like in case of boardroom voting, has significantly larger effect on the mix net approach than on homomorphic tallying approaches. Furthermore, the precomputations have a significant effect on the overall efficiency of mix net approach, which tends to be higher, especially with a high total number of eligible voters.

The homomorphic tallying approaches tend to be less efficient as ballot complexity increases. If the vote is encoded in a single ciphertext, the length of the

exponent representing the election result strongly depends on both number of voters and candidates, while the proofs themselves remain relatively efficient, which is why this approach outperforms others in case of simple "yes/no" elections. In the homomorphic tallying approaches that require encoding the vote in multiple ciphertexts, while the exponent size remains relatively small even with the larger number of candidates and voters, the amount and the complexity of validity proofs to be constructed and verified per vote, becomes larger, thus making the election less efficient.

Table 10. Evaluation results of different settings

Election	Election stage	Mixnet	HT: Helios	HT: Groth	HT: Joachim
Approval voting: Estonia	Precomputations	20.09 h.	**2.061 s.**	> 30 days	190.4 h.
	Voting	**0.009 s.**	14.97 s.	0.024 s.	21.4 s.
	Validation	**14.08 m.**	441.1 h.	31.83 m.	> 30 days
	Tallying	2.316 h.	22.78 s.	**0.039 s.**	22.78 s.
Approval voting: Norway	Precomputations	4.422 h.	**0.6 s.**	> 30 days	1.81 m.
	Voting	**0.009 s.**	0.411 s.	0.024 s.	0.603 s.
	Validation	**2.774 m.**	2.366 h.	6.269 m.	5.323 h.
	Tallying	37.22 m.	1.185 s.	**0.06 s.**	1.185 s.
Approval voting: Switzerland	Precomputations	1.574 h.	**0.495 s.**	8.976 s.	1.647 s.
	Voting	**0.009 s.**	0.018 s.	0.021 s.	0.057 s.
	Validation	**1.405 m.**	3.344 m.	2.775 m.	18.96 m.
	Tallying	9.239 m.	**0.03 s.**	0.03 s.	0.03 s.
Approval voting: IACR	Precomputations	1.225 m.	**0.387 s.**	> 30 days	4.71 s.
	Voting	**0.009 s.**	0.096 s.	0.024 s.	0.216 s.
	Validation	**3.108 s.**	36.98 s.	7.32 s.	2.449 m.
	Tallying	16.55 s.	0.084 s.	**0.021 s.**	0.084 s.
Approval voting: Boardroom	Precomputations	8.433 s.	**0.384 s.**	3.097 m.	3.846 s.
	Voting	**0.009 s.**	0.108 s.	0.021 s.	0.156 s.
	Validation	**0.174 s.**	2.484 s.	0.345 s.	5.508 s.
	Tallying	7.887 s.	0.786 s.	**0.159 s.**	0.786 s.
Divisive voting	Precomputations	1.342 h.	**1.617 s.**	-	16.98 m.
	Voting	**0.009 s.**	17.01 s.	-	28.23 s.
	Validation	**4.438 m.**	183.1 h.	-	474.9 h.
	Tallying	23.54 m.	**6.84 s.**	-	6.84 s.
Ranking voting	Precomputations	1.5 m.	-	> 30 days	**38.59 s.**
	Voting	**0.009 s.**	-	0.045 s.	33.86 s.
	Validation	**6.726 s.**	-	38.2 s.	11.83 h.
	Tallying	1.108 m.	-	**0.039 s.**	47.34 s.

6 Conclusion

We have evaluated different approaches to anonymize the votes in different settings with regards to their efficiency. Namely, we focused on two anonymisation

approaches common in use: mix net and homomorphic tallying with different ballot types. Furthermore, we have built a prototype of a tool that enables election organizers to perform such a comparison themselves with regards to their chosen setting, in order to choose the most efficient approach. As we found out, there is no single approach that is the most efficient in all the possible election settings. Therefore, an individual evaluation has to be done for each setting, which is what our tool is designed to assist in.

While efficiency is an important consideration in implementing e-votin systems, there are other criteria that can suggest using one approach over another. In particular, revealing individual votes, which is inavoidable in mixnet-based approach, can lead to privacy issues in certain settings, for example, in very small-scale elections, or if coercion and vote buying is an issue. Thus, in case both homomorphic tallying and mixnet-based approach is available for certain kind of elections, the organizers have to evaluate themselves the trade-off between efficiency and possible privacy concerns, while deciding for one or another approach. Identifying the scenarios, in which such privacy issues can arise, is the question of future work.

Acknowledgment

This project (HA project no. 435/14-25) is funded in the framework of Hessen ModellProjekte, financed with funds of LOEWE – Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz, Förderlinie 3: KMU-Verbundvorhaben (State Offensive for the Development of Scientific and Economic Excellence).

References

1. Adida, B.: Helios: Web-based open-audit voting. In: USENIX Security Symposium. vol. 17, pp. 335–348 (2008)
2. Adida, B., De Marneffe, O., Pereira, O., Quisquater, J.J., et al.: Electing a university president using open-audit voting: Analysis of real-world use of helios. EVT/WOTE 9, 10–10 (2009)
3. Burton, C., Culnane, C., Schneider, S.: Secure and verifiable electronic voting in practice: the use of vvote in the victorian state election. arXiv preprint arXiv:1504.07098 (2015)
4. Committee, E.N.E.: Riigikogu elections 2011 - riigikogu (parliament) elections - past elections - estonian national electoral committee. http://www.vvk.ee/past-elections/riigikogu-parliament-elections/riigikogu-elections-2011/ (2011), [Online; accessed 2-March-2015]
5. Committee, E.N.E.: Statistics - internet voting - voting methods in estonia - estonian national electoral committee. http://www.vvk.ee/voting-methods-in-estonia/engindex/statistics (2015), [Online; accessed 2-March-2015]
6. Culnane, C., Ryan, P.Y., Schneider, S., Teague, V.: vvote: a verifiable voting system. arXiv preprint arXiv:1404.6822 (2014)
7. Dubuis, E., Fischli, S., Haenni, R., Hauser, S., Koenig, R.E., Locher, P., Ritter, J., von Bergen, P.: Verifizierbare internet-wahlen an schweizer hochschulen mit univote. In: GI-Jahrestagung. pp. 767–788. Citeseer (2013)

8. Gjøsteen, K.: The norwegian internet voting protocol. In: E-voting and identity, pp. 1–18. Springer (2012)
9. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Applied Cryptography and Network Security. pp. 467–482. Springer (2005)
10. Hao, F., Ryan, P.Y., Zieliński, P.: Anonymous voting by two-round public discussion. IET Information Security 4(2), 62–67 (2010)
11. Helios: Helios v4. http://documentation.heliosvoting.org/verification-specs/helios-v4 (2012), [Online; accessed 2-March-2015]
12. IACR: IACR Election 2012. http://www.iacr.org/elections/2012/ (2012), [Online; accessed 2-March-2015]
13. Joaquim, R.: How to prove the validity of a complex ballot encryption to the voter and the public. Journal of Information Security and Applications 19(2), 130–142 (2014)
14. Jordi Barrat i Esteve, B.G., Turner, J.: Norwegian E-vote Project - Speed and Efficiency of the Vote Counting Process. `https://www.regjeringen.no/globalassets/upload/krd/prosjekter/e-valg/evaluering/topic4_assessment.pdf` (2012), [Online; accessed 2-March-2015]
15. Kiayias, A., Yung, M.: Self-tallying elections and perfect ballot secrecy. In: Public Key Cryptography. pp. 141–158. Springer (2002)
16. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC press (1996)
17. Organization for Security and Co-operation in Europe: Switzerland, Federal Elections, 23 October 2011: Final Report. `http://www.osce.org/odihr/87417` (2015), [Online; accessed 30-October-2015]
18. Pedersen, T.P.: Distributed provers and verifiable secret sharing based on the discrete logarithm problem. DAIMI Report Series 21(388) (1992)
19. Schweizerische Bundeskanzlei: Vote électronique - Versuchübersicht. `https://www.bk.admin.ch/themen/pore/evoting/08004/index.html?lang=de` (2015), [Online; accessed 30-October-2015]
20. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: Progress in Cryptology–AFRICACRYPT 2010, pp. 100–113. Springer (2010)
21. Volkamer, M., Budurushi, J., Demirel, D.: Vote casting device with vv-sv-pat for elections with complicated ballot papers. In: Requirements Engineering for Electronic Voting Systems (REVOTE), 2011 International Workshop on. pp. 1–8. IEEE (2011)
22. Wikström, D.: A commitment-consistent proof of a shuffle. In: Information Security and Privacy. pp. 407–421. Springer (2009)
23. Wikström, D.: How to implement a stand-alone verifier for the verificatum mix-net (2011)