Tatra Mt. Math. Publ. **41** (2008), 93–106

TATRA MOUNTAINS
Mathematical Publications

SINGLE TRANSFERABLE VOTE ANALOGUE OF DESMEDT-KUROSAWA VOTING PROTOCOL

Joanna Boroń — Marek Klonowski

ABSTRACT. During Information Security Conference 2005, Yvo Desmedt and Kaoru Kurosawa presented an electronic voting scheme that allows to point the winner, without revealing the final tally — that is, no information is disclosed except for the name of the winner(s). In particular, even the number of votes for every single candidate remains hidden. It can be important in practice, since these numbers can reveal sensitive information about voters' preferences in some scenarios, especially in closed, small communities. In some sense, this scheme meets stronger privacy requirements than any voting protocol before.

The main contribution of our paper is a version of Desmedt-Kurosawa scheme that supports Single Transferable Vote (STV) used among others in Republic of Ireland and Australia.

The presented solution has the main properties of Desmedt-Kurosawa scheme, however we had to solve several subtle problems in order to implement the idea of SVT that is much more complicated than the regular voting. For example, we have even to hide the number of rounds that does not need to be constant. As in the original scheme, every step can be publicly verified.

1. Introduction

Idea of voting using electronic devices and/or electronic networks gained recently considerable attention in both research community as well as in state authorities in many countries. Most of designed schemes are focused either on providing additional features to hand-counted ballot procedure (see, e.g., [2, 6, 3, 14, 15]), securing voting machines (see, e.g., [10]) or designing schemes, where voting is possible over electronic networks (see, e.g., [11]). One of the most important features is verifiability — possibility to check that each vote has been counted and that the results are correctly computed. Desirable are solutions wherein any party can verify correctness of the result using published data.

2000 Mathematics Subject Classification: 94A60, 11T71.
Keywords: e-voting, single transferable vote.
Partially supported by Polish Ministry of Science and Higher Education, grant N206 2701 33.

While the papers mentioned above can be regarded as electronic versions of traditional voting scenarios, D e s m e d t and K u r o s a w a proposed slightly different approach [5]. Namely, they designed the very first voting scheme that reveals noting except the name of the winner. In practice it means, that after the execution of the procedure no one knows the number of votes cast for the winner or names of the candidates with the second, the third etc. position.

Spirit of these scheme is in some sense similar to electronic-auctions protocols. Paper [5] presents situations, for which information such as the number of votes supporting the winner are quite sensitive. In some sense Desmedt-Kurosawa scheme meets possibly most demanding privacy requirements — i.e., the name of the winner is revealed and nothing more. Thereby, only the information that MUST be revealed in order to accomplish the goal of the voting procedure is published. Security of this scheme as a whole has not been proved, but it is based only on provably secure cryptographic primitives.

In this paper we present an extension of the Desmedt-Kurosawa scheme for Single Transferable Vote procedures. Single Transferable Vote (STV) is a voting system designed in order to reduce the effect of *wasted votes* (e.g., the votes cast for parties that finally do not pass the threshold necessary to get any seat in the parliament). The goal of STV is to provide proportional representation while ensuring that votes are cast for candidates rather than party lists [17]. STV is currently used during some elections in Australia, Scotland and Northern Ireland.

Aside Desmedt-Kurosawa's idea, we also reuse several techniques presented in their paper [5]. However, our task is more challenging, since STV system is conceptually much more complicated than a regular voting scheme. STV procedure executes several phases and only the number of phases reveals information about voting statistics. In particular, the number of necessary phases is unknown beforehand.

Of course, our scheme, exactly like the original one, does not use any trusted third party. With a help of a TTP building such scheme would be a trivial task, but then the fundamental principle of leaking no additional information to any protocol participant would be obviously violated.

1.1. Paper organization

In Section 2 we present details of STV procedure. In Section 3 we introduce all cryptographic primitives and procedures used in the scheme description that is set forth in Section 4. We estimate the efficiency of our scheme in Section 5.

2. Single transferable vote

2.1. Why STV?

In commonly used voting systems a voter often votes tactically and chooses just one of the two leading candidates even if none of them is his or her favorite one. A vote cast for a candidate who is not very popular and is likely to loose the election can be often considered wasted.

In STV voters have to rank all candidates by the order of their preference rather than to choose only a single candidate. The voters' preferences are considered in a wider scope, so one can say that election results more accurately express electors' will.

In this paper we consider a single-winner SVT election. This variant of STV is also known as *Instant-runoff* voting (IRV) or *Alternative Vote* (AV).

2.2. STV algorithm

A voter ranks all candidates according to his or her preferences. Each ballot contains then a permuted list of all candidates with the most favoured candidate in the first position, the second favourite candidate in the second one, and so on. When all ballots are collected in the ballot-box, the following steps are executed:

(1) The first position in each ballot is taken into account. If there is a candidate who gets the majority, he is declared to be the winner and the procedure is terminated.
(2) The candidate with the lowest number of votes is eliminated and deleted from all ballots. If there are two or more candidates who get the lowest number of votes, one of them is chosen according to some further rules.
(3) Go to step 1.

One can easily see that during each round either the winner is pointed or one candidate is eliminated. For that reason the number of possible rounds is smaller than the number of the candidates. Let us also stress that the number of rounds executed during single election cannot be determined in advance and is strongly dependent on submitted ballots.

Example. We show on an example how STV works. Assume that there are 24 voters and 4 candidates: A, B, C, D. The first table shows how many voters marked specific order of candidates in their ballots.

	Votes				
	6	6	2	7	3
1st	D	A	B	C	B
2nd	A	B	A	A	C
3rd	C	C	C	B	A
4th	B	D	D	D	D

The second table shows process of counting the result round by round.

	round 1:					round 2:					round 3:				
		6	6	2	7	3		6	6	2	7	3		6	6	2	7	3
Current ballots	1st	D	A	B	C	B	1st	D	A	A	C	C	1st	A	A	A	C	C
	2nd	A	B	A	A	C	2nd	A	C	C	A	A	2nd	C	C	C	A	A
	3rd	C	C	C	B	A	3rd	C	D	D	D	D						
	4th	B	D	D	D	D												
# votes for A	6						8						14					
# votes for B	5																	
# votes for C	7						10						10					
# votes for D	6						6											

At the first round candidate B received the fewest number of votes and is eliminated. His votes are transferred to other candidates, in this case 3 votes go to C and 2 votes go to A. In consequence in the second round D becomes the weakest candidate and gets eliminated. After the third round candidate A gains 14 votes which gives him majority and counting terminates.

2.3. Tie breaking

A tie can occur when selecting a candidate to eliminate. There are many methods to break the tie using not only information from the current round. One can take into account the result of previous rounds or, which is more complicated, try to simulate eliminating each of the candidates in question and take into account later rounds.

If a tie can be broken with established rules, we call it a *weak tie*. But sometimes the tie remains not broken, even after the tie-breaking rules have been applied. In that case it is a *strong tie*. When a strong tie occurs, it is impossible to choose between two or more candidates. In such case it is better to break a strong tie by lot and eliminate always only one candidate instead of removing several candidates (see example below).

Our scheme uses backwards tie-breaking method. In this approach, if there are two or more candidates who have the minimal number of votes, we choose among them the candidate who had fewer votes during the previous round.

If the number of votes is also equal in previous round, the latest point in the count where they had unequal number of votes is taken into account.

The strong tie occurs when candidates have equal number of votes in all previous rounds. Our scheme breaks strong ties by lot, but it can be adapted in obvious manner to most of the rules for breaking a strong tie considered so far.

A comparison of some tie-breaking methods and advantages of backwards tie-breaking can be found in [13].

EXAMPLE. We show an example of a strong tie such that choosing a candidate for elimination at random is better then rejecting more than one candidate. The following table presents distribution of votes:

	Number of Votes			
	10	10	6	6
1st	B	A	C	C
2nd	A	B	A	B
3rd	C	C	B	A

In the first round there is a tie between candidates A and B. What is more, these candidates have the smallest number of votes from first preferences, which qualifies them for elimination. After rejecting both A and B only, the candidate C remains and becomes the winner. On the other hand, from among 32 voters 20 voters prefer whichever A or B rather than C so this result definitely does not express voters' will well. In this case, it is better to eliminate only one of A and B and thus to let B or A win. Since here A and B received exactly the same number of votes, even taking into account all preferences, the only method to choose between them is a lot.

3. Cryptographic tools

To implement the idea of STV in electronic form, we will need several cryptographic tools described in this section.

Private keys used in the scheme are shared by N authorities. For all operations mentioned below we need robust threshold versions, i.e., when at most t of authorities (where $N \geq 2t + 1$) are malicious, the protocol still works correctly. In some sense authorities jointly simulate a trusted third party. Let us stress that there are many cryptographic schemes meeting this requirement.

3.1. Encryption functions

The most important building block for our scheme are two efficient, probabilistic public-key encryption functions with homomorphic property. We have two types of data to encrypt: candidates and numbers. "Candidate" means here, for example, a result of a hash function on the candidate's name representing real person.

For encryption of these values we use function F. Other data are numbers and we use function E to encrypt it. The reason why we use two different functions is that the encryption algorithm used for numbers must satisfy some additional requirements, but on the other hand, we can assume that plaintext can be only a number from a relatively small set.

Let us use the following notation:

- $E(m)$, $F(m)$ are the sets of possible ciphertexts for a plaintext m, for functions E and F, respectively.
- $E_0(m)$, $F_0(m)$ are ciphertexts of m using the random parameter set to 0. We call these ciphertexts *standard ciphertexts* of m. Let us note that everyone can easily check if a particular ciphertext is the standard ciphertext of a message m.
- $e_1 =_E e_2$, $e_1 =_F e_2$ mean that e_1 and e_2 are ciphertexts of the same plaintext using function E and F, respectively.

The functions E and F must meet additional requirements. Namely, they must provide homomorphic property:

There are computationally efficient operations $^{-1}$ and $\odot$, such that

(1) if $e \in F(m)$, then $e^{-1} \in F(m^{-1})$,

(2) if $e_1 \in F(m_1)$ and $e_2 \in F(m_2)$, then $e_1 \odot e_2 \in F(m_1 m_2)$.

Function E has additive homomorphic property: there exist effectively computable operations $\ominus$ and $\oplus$, such that

(1) if $e \in E(m)$, then $\ominus e \in E(-m \bmod q)$,

(2) if $e_1 \in E(m_1)$ and $e_2 \in E(m_2)$, then $e_1 \oplus e_2 \in E(m_1 + m_2 \bmod q)$

for some large q. Let us note that the second property allows to sum up the encrypted number of votes without decryption of the corresponding ciphertexts. Indeed — product of ciphertexts of two values is a ciphertext of the sum of these values.

Such properties allow also to perform re-encryption. Given $e \in E(m)$ (or $e \in F(m)$) one can compute e' such that $e =_E e'$ (or $e =_F e'$, respectively) where e' is uniformly distributed over $E(m)$ (respectively, $F(m)$). It can be done easily by computing a ciphertext of the neutral element in appropriate underlying group and applying homomorphic operation on this ciphertext and e.

Candidates for functions F and E. Let us note that regular ElGamal encryption scheme over a group of order q can be used as the function E. As pointed in [5] modified ElGamal may play a role of the function F, i.e.:

$$F_r(m) = (g^r, g^m y^r) \mod p,$$

where p is a large prime, y is a public key and g is a group generator. Further details can be found in [5].

3.2. Plaintext Equality Test

We use Plaintext Equality Test (PET) on ciphertexts of E and F. Using PET procedure, N authorities given two ciphertexts e_1 and e_2 can jointly check whether $e_1 =_E e_2$ (or $e_1 =_F e_2$, respectively) without revealing any other information. For this purpose one can use, for instance, the procedure from [7]. Protocol presented in this paper is based on ElGamal encryption. However, one can build PET procedure for any function F with properties mentioned above.

3.3. 1-out-of-n re-encryption proof

During 1-out-of-n re-encryption proof a prover shows that for an encrypted message e and a set of ciphertexts $e_1, e_2, \ldots, e_n$, there exists an index $1 \leq i \leq n$, such that $e =_E e_i$ (or $e =_F e_i$) without revealing any other information. In particular, the index i remains secret. It is assumed that the prover knows randomness used for re-encryption.

Description of such a proof can be found for example in [4].

3.4. MIX protocol

In our protocol we extensively use MIX protocol on ciphertexts computed with function E or F. Using the MIX protocol, N authorities can jointly compute permutation $\left(e'_{\pi(1)}, \ldots, e'_{\pi(n)}\right)$ of ciphertexts list $(e_1, \ldots, e_n)$, such that $e'_{\pi(i)}$ is a re-encryption of e_i for every i. The validity of this operation can be publicly verified, but the whole procedure does not reveal any unnecessary information. In particular, the permutation π remains hidden. In our scheme (as well as in [5]) we need a version of MIX which allows to permute and re-encrypt list of pairs of ciphertexts but without breaking particular pairs. That is, for given list $\big((e_1, f_1), \ldots, (e_n, f_n)\big)$ the authorities compute a list of permuted re-encryptions $\big((e'_{\pi(1)}, f'_{\pi(1)}), \ldots, (e'_{\pi(n)}, f'_{\pi(n)})\big)$. Such a MIX protocol can be built, for example, using algorithms from papers [8, 9]. Let us stress that this protocol is jointly perforemed by N authorities in such a way that any cooperating coalition of t (for $N \geq 2t + 1$) entities is not able to learn π.

3.5. Pairwise comparison protocol

In the scheme (called PCP) described in the next section we need very often to compare two numbers encrypted with function E without decrypting them.

D e s m e d t and K u r o s a w a proposed in [5] a way to compare two encrypted numbers in such way that it can be publicly verified and nothing more than comparison result is revealed.

Given $A \in E(a)$ and $B \in E(b)$ the protocol decides whether $a \leq b$ or $a > b$. The numbers a and b must satisfy $0 \leq a \leq M$, $0 \leq b \leq M$ for some M. For the sake of efficiency, M should be as small as possible. In our scheme we use comparison for number of votes multiplied by number of candidates, so $0 \leq a$, $b \leq vk$, where v is the number of voters and k is the number of candidates.

4. Voting scheme

There are N authorities representing for example political parties or particular candidates. Let us also assume that at most t of them are malicious (where $N \geq 2t+1$). So in particular, majority of authorities operates exactly according to the protocol. We assume that their public keys are widely known.

All operations are preformed *jointly* by authorities. It means in some sens that they simulate a trusted third party that performs some operations on behalf of them. Thanks to this property some details of protocols execution remain hidden for single authorities or even coalition of cardinality less or equal t.

We also need a *bulletin board* — it is a shared memory, that everyone can read, and also can append some information (without deleting the previous notes). We assume that transcript of all subprocedures are placed there. Let us also stress that it is a standard requirement in such a type of protocols.

We assume that there are v voters and k candidates, say $C_1, C_2, \ldots, C_k$.

The whole voting process is divided into two stages. During the first one all voters prepare their electronic ballots. In the second stage during $k-1$ steps authorities are pointing the winner.

4.1. Votes casting

(1) A voter i sort the list of candidates with respect to his own preference: $C_1^i, C_2^i, \ldots, C_k^i$, where C_1^i is the best candidate in ith voter's opinion.

(2) He computes ciphertexts $c_1^i \in F(C_1^i), \ldots, c_k^i \in F(C_k^i)$ and posts them on the bulletin board.

(3) Validity of $c_1^i, \ldots, c_k^i$ is shown as follows:
- The voter proves that each of $c_1^i, \ldots, c_k^i$ encrypts one of the candidates C_1, C_2,...,C_k by performing k times 1-out-of-k re-encryption proof.
- The authorities jointly check whether $c_1^i, \ldots, c_k^i$ are ciphertexts of distinct values using plaintext equality test.

After this step, authorities can be sure that the tuple $c_1^i, \ldots, c_k^i$ is correctly prepared — it represents an encrypted list of all, distinct candidates.

(4) The tuple $c_1^i, \ldots, c_k^i$ is transformed to the electronic ballot:

$$V_i = \left(\left(E_0(1), c_1^i \right), \ldots, \left(E_0(k), c_k^i \right) \right).$$

4.2. Computing election result

Counting proceeds in rounds as for STV but there are always $k-1$ rounds. Admittedly, sometimes a majority is achieved after less than $k-1$ rounds and the procedure in the real world scenario may terminate, but we want to hide the number of rounds. This has no impact in our algorithm on pointing the winner, since once a candidate achieves majority he will never be eliminated and will remain in the game until the end of the last round.

4.2.1. Round

During each round, the votes from the first preference in each ballot are taken into account. Then the candidates are sorted according to the number of votes and the candidate with the fewest votes is eliminated from every ballot. After that in every ballot from the remaining preferences the highest one is marked as the first preference.

Computation operates on a list L and lists V_i (electronic ballots) for every voter i. The list L is initialized in every round afresh, when V_i's are modified in each round. The list L stores pairs (candidate, number of votes for this candidate), where both values are encrypted. Additionally, the number of votes is stored not as the exact value but is multiplied by the number of candidates, because this value modulo number of candidates contains information about position in the previous round. In general, one can think of it as a number of votes for the candidate because the position from previous round is significant only in case of a tie.

As stated above, the list V_i is a ballot of voter i. It contains pairs (index of preference, candidate) as at the last step of vote casting. In every round one candidate is removed and order of all V_i is changed.

At the beginning of this stage authorities jointly permute and re-encrypt at random the list $F_0(C_1), F_0(C_2), \ldots, F_0(C_k)$ and let $c_1, c_2, \ldots, c_k$ be the resulting list of ciphertexts.

The authorities execute the following steps for $r = 1, \ldots, k-1$. At the beginning of round r there are $p = k - r + 1$ candidates left and exactly one of them will be eliminated in the round. The procedure starts with all k candidates and before the last round there are only 2 candidates.

(1) Let $c_1, c_2, \ldots, c_p$ be ciphertexts of the candidates in the order from the previous round. (Note that $p = k - r + 1$). Initiate L as

$$\mathrm{MIX}\big(c_1, E_0(p-1)\big), \Big(c_2, E_0(p-2), \ldots, \big(c_p, E_0(0)\big)\Big).$$

In this way candidates are strictly and randomly ordered before counting votes for this round and this prevents the risk of a tie. If there is a method to break a strong tie, the candidates could be ordered according to special rules also before the first round.

(2) For every voter i:
- Let c be the second element of the first pair of V_i. The first pair is the first preference, so c is a ciphertext of the most preferred candidate (of those that remain in the game) of voter i.

1: Find on the list L element (c', t) such that $c =_F c'$.
2: Increase votes for candidate c by changing the pair (c, t) to a pair $\big((c, t \oplus E_0(p)\big)$.
3: Mix L up before considering the next ballot.

The first step is implemented by trying plaintext equality test on each element of the list. Let us note that the votes are counted p-fold because initially every candidate gains a number of "votes" corresponding to the position immediately after the previous round. This trick solves of a tie. Indeed, in case of a tie in particular round position from previous round(s) decides about the order. The third step is implemented by the MIX protocol.

(3) Sort the list L by the second element of every pair. The pairwise comparison protocol (PCP) is used here. Sorting can be implemented by any algorithm that is based on comparing pairs of elements to be sorted only.

(4) Let e be the first element of the last pair on list L. Then e is a ciphertext of the candidate to be eliminated. Remove from L the pair containing e. Retain the current ordering of the candidates for the next round.

(5) For every voter i

1: Mix V_i up.
2: Find in V_i a pair (p, c) such that $c =_F e$ and remove this pair from V_i.
3: Find in V_i a pair (p', c') with the smallest first element and move it to the first position of V_i.

In the first point MIX protocol is used. The second point is based on plaintext equality test. The pairwise comparison protocol is used for finding the minimum value in the third point. Let us note that there is no need to sort whole V_i, since only the highest preference are counted.

Since at each round exactly one candidate is removed, after $k-1$ rounds there is only one candidate and this one is the winner. The authorities jointly decrypt the ciphertext containing his name and announce it.

Let us note that this protocol is *oblivious* — i.e., from observer's point of view its execution looks exactly the same, independently of voters' preferences. On the other hand, at the end of the protocol, only one candidate is on the list.

5. Efficiency

Efficiency of the scheme depends on many parameters, but here we consider only the number of voters (denoted by v) and candidates (denoted by k). We assume that the remaining parameters, e.g., the number of authorities and the length of keys using for encryption are predetermined.

Since our scheme uses many cryptographic operations, we first introduce symbols for the time complexity of these operations:

- T_{enc} — encryption using E or F,
- $T_{\circ}$ — homomorphic binary operation on ciphertexts,
- T_{PET} — plaintext equality test (PET),
- $T_{\text{1-of}}(n)$ — 1-out-of-n re-encryption proof,
- $T_{\text{MIX}}(n)$ — MIX protocol on a list of the length n,
- $T_{\text{PCP}}(n)$ — pairwise comparison protocol on numbers less or equal to n.

Most functions and protocols that we use are commonly known and can be found in different versions with different time complexities so we do not use specific values. We only mention efficiency of comparison algorithm proposed by Desmedt and Kurosawa in [5]. For $0 \leq a, b \leq n$ the protocol decides whether $a \leq b$ using n homomorphic operations, applying $n+1$ times PET procedure and performing MIX on list of length $n+1$. Thus

$$T_{\text{PCP}}(n) = O(n) \times (T_{\circ} + T_{\text{PET}}) + T_{\text{MIX}}(n).$$

Roughly estimated complexities of particular steps of vote casting are listed below:

- step 1: $O(1)$,
- step 2: $O(k) \times T_{\text{enc}}$,
- step 3: $O(k) \times T_{\text{1-of}}(k) + O\big(k^2\big) \times T_{\text{PET}}$,
- step 4: $O(k) \times T_{\text{enc}}$.

Let us note that $T_\circ$, T_{enc} and T_{PET} do not depend on number of candidates and voters so we can consider them as constant time operations. Validation of a vote (step 3) is the most consuming step so efficiency of the whole vote casting stage is equal to

$$O(k) \times T_{\text{1-of}}(k) + O\big(k^2\big)$$

and this operation must be done for every voter.

The second stage of the scheme is computing the result which proceeds in $O(k)$ rounds. Below we list time complexities of particular steps of one round:

- step 1: $O(k) \times T_{\text{enc}} + T_{\text{MIX}}(k)$,
- step 2: $v \times O(k) \times T_{\text{PET}} + O(1) \times (T_\circ + T_{\text{enc}}) + T_{\text{MIX}}(k)$,
- step 3: $O(k \log k) \times T_{\text{PCP}}(vk)$,
- step 4: $O(1)$,
- step 5: $v \times T_{\text{MIX}}(k) + O(k) \times T_{\text{PET}} + O(k) \times T_{\text{PCP}}(vk)$.

So the final cost of this stage is

$$O\big(vk^2\big) \times T_{\text{PCP}}(vk) + O\big(k^2 \log k\big) \times T_{\text{PCP}}(vk) + O(vk) \times T_{\text{MIX}}(k).$$

Hence, the overall time of the protocol is

$$O\big(vk^2\big) \times T_{\text{PCP}}(vk) + O\big(k^2 \log k\big) \times T_{\text{PCP}}(vk) + O(vk) \times T_{\text{MIX}}(k) + O(k) \times T_{\text{1-of}}(k).$$

Assuming

$$T_{\text{PCP}}(vk) = O(vk), T_{\text{MIX}}(k) = O(k), T_{\text{1-of}}(k) = O(k),$$

we get the final asymptotic complexity of the protocol equal to

$$O\big(v^2k^3 + vk^3 \log k\big).$$

6. Open problems and closing remarks

We presented a scheme that should work well for small communities. We are in quite convenient situation, since motivation for hiding absolutely all information except the name of the winner is mostly desirable for small and closed groups of people, wherein some details can reveal preferences of individuals. However, it seems interesting to propose a scheme that could be much more efficient and applicable for grater number of voters even for the price of revealing some details.

Another open problem is to build a similar scheme that could support multi-seat as well as "write-in" elections.

Acknowledgements. We would like to express thanks to Professor Mirosław K u t y ł o w s k i and Professor Przemysław B ł a ś k i e w i c z for their valuable comments.

REFERENCES

[1] CHAUM, D.: *Untraceable electronic mail, return addresses, and digital pseudonyms* Commun. ACM **24** (1981), 84–90.

[2] CHAUM D.: *Secret-ballot receipts and transparent integrity,* `http://www.vreceipt.com/article.pdf`.

[3] CHAUM, D.—RYAN, P.—SCHNEIDER, S.: *A practical voter-verifiable election scheme,* in: Computer Security—ESORICS '05, Lecture Notes in Comput. Sci., Vol. 3679, Springer--Verlag, Berlin, 2005, pp. 118–139.

[4] CRAMER, R.—GENNARO, R.—SCHOENMAKERS, B.: *A secure and optimally efficient multi-authority election scheme,* in: Advances in Cryptology—EUROCRYPT '97 (W. Fumy, ed.), Lecture Notes in Comput. Sci., Vol. 1233, Springer-Verlag, Berlin, 1997, pp. 103–118.

[5] DESMEDT, Y.—KUROSAWA, K.: *Electronic voting: starting over?* in: Information Security Conference—ISC '05 (J. Zhou et al., eds.), Lecture Notes in Comput. Sci., Vol. 3650, Springer-Verlag, Berlin, 2005, pp. 329–343.

[6] HOSP, B.—POPOVENIUC, S.: *An introduction to punchscan,* in: Workshop on Rating Voting Methods—VSRW '06, 2006, `http://vote.cs.gwu.edu/vsrw2006/papers/9.pdf`.

[7] JAKOBSSON, M.—JUELS, A.: *Mix and match: secure function evaluation via ciphertexts,* in: Advances in Cryptology—ASIACRYPT '00 (T. Okamoto, ed.), Lecture Notes in Comput. Sci., Vol. 1976, Springer-Verlag, Berlin, 2000, pp. 162–177.

[8] JAKOBSSON, M.—JULES, A.: *An optimally robust hybrid mix network,* in: Proc. of the 20th Annual ACM Symposium on Principles of Distributed Computing—PODC '01, ACM, New York, 2001, pp. 284–292.

[9] JAKOBSSON, M.—JUELS, A.—RIVEST, R. L.: *Making mix nets robust for electronic voting by randomized partial checking,* in: Proc. of the 11th USENIX Security Symposium (D. Boneh, ed.), USENIX Association, Berkeley, CA, USA, 2002, pp. 339–353.

[10] KLONOWSKI, M.—KUTYŁOWSKI, M.—LAUKS, A.—ZAGÓRSKI, F.: *A practical voting scheme with receipts,* in: Information Security—ISC '05 (J. Zhou et al., eds.), Lecture Notes in Comput. Sci., Vol. 3650, Springer-Verlag, Berlin, pp. 490–497.

[11] KUTYŁOWSKI, M.—ZAGÓRSKI, F.: *Verifiable internet voting solving secure platform problem,* in: 2nd International Workshop on Security—IWSEC '07 (A. Miyaji et al., eds.), Lecture Notes in Comput. Sci., Vol. 4752, Springer-Verlag, Berlin, 2007, pp. 199–213.

[12] NEFF, C.A.: *A Verifiable secret shuffle and its application to E-voting,* in: Proc. of the 8th ACM Conference on Computer and Communications Security—ACM CCS '01 (P. Samarati, ed.), ACM Press, New York, USA, 2001, pp. 116–125.

[13] O'NEILL, J.: *Tie-breaking with the single transferable vote,* Voting matters **18** (2004), 14–17.

[14] RIVEST, R. L.: *The threeballot voting system,* Draft Version 10/1/06 2006.

[15] RIVEST, R. L.—SMITH, W.: *Three voting protocols: ThreeBallot, VAV, and Twin,* in: Proc. of USENIX/ACCURATE Electronic Voting Technology Workshop—EVT '07, USENIX Association, Berkeley, CA, USA, 2007.

[16] VAN DE GRAAF, J.: *Adapting Chaum's voter-verifiable election scheme to the Brazilian system.* http://www.ppgia.pucpr.br/~maziero/pesquisa/ceseg/wseg04/2958.pdf.

[17] COLOMER, J. M. (ed.): *Handbook of Electoral System Choice.* Palgrave-Macmillan, London, 2004.

Received September 29, 2007

Joanna Boroń
Brains on Wings
Pl. Strzelecki 20
PL-50–224 Wrocław
POLAND
E-mail: asia@gorska.net

Marek Klonowski
Institute of Mathematics and
Computer Science
Wrocław University of Technology
ul. Wybrzeże Wyspiańskiego 27
PL-50–370 Wrocław
POLAND
E-mail: marek.klonowski@im.pwr.wroc.pl