JETS

The USENIX Journal of Election Technology and Systems

Volume 2, Number 1 • December 2013

usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

JETS

The USENIX Journal of Election Technology and Systems

Volume 2, Number 1 • December 2013

JETS articles will be presented at the Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE).
www.usenix.org/conferences/evtwote

ISBN 978-1-931971-07-2 ISSN 2328-2797

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

USENIX Journal of Election Technology and Systems (JETS)
Volume 2, Number 1 • December 2013
www.usenix.org/jets/issues/0201

Do new voting technologies prevent fraud? Evidence from Russia

MAX BADER, Leiden University

Widespread concerns exist that new voting technologies invite electoral fraud. In states with a known record of electoral fraud, however, the use of new voting technologies may help reduce the incidence of fraud by automating parts of the voting and counting process. This study shows that the use of optical scan voting systems had a significant effect in terms of fraud reduction during the 2011 legislative election in Russia. This finding has implications for organizations and governments that seek to promote democratic elections in undemocratic states..

1. INTRODUCTION

While new voting technologies become more widespread across the world, a number of countries in recent years have rolled back the use of these technologies [Goldsmith 2011: 1]. When this happens, the reason in most cases is related to concerns that the technologies are vulnerable to voting fraud [e.g. Loeber 2008; Volkamer 2010]. New voting technologies in electoral authoritarian regimes, however, can play a role in reducing the incidence of fraud because they limit human involvement in the voting and counting process. Recent reports that the Central Election Commission of Russia is abandoning the use of new voting technologies, therefore, are thought to increase rather than decrease opportunies for voting fraud in that country [Buzin 2013; Sergeeva 2013].

There is much anecdotal evidence - but not more than that - that the use of optical scan voting systems in Russia over the past decade is associated with reduced fraud in the polling stations where these systems are used [e.g. Kobak 2011; Pshenichnikov 2011]. The evidence is usually based on a direct comparison of election results from polling stations where the optical scan voting systems were used, with election results from polling stations where these systems were not used. The evidence is unsatisfactory because the optical scan voting systems are not randomly distributed [Shen' 2012]. This study instead uses a differences-in-differences design to assess the claim that optical scan voting systems in Russia have reduced fraud: rather than comparing the results of polling stations that used optical scan voting systems with the results of polling stations that did not use them, we compare the relative election results - relative to election results of polling stations in the same district – from the December 2011 legislative election and 2012 presidential election in Russia of polling stations that did use optical scan voting systems in one of the two elections but not in the other. The article proceeds in three sections. The first section discusses the use of new voting technologies in Russian elections. The second section discusses what we know about fraud in the 2011 and 2012 Russian elections. The third section, finally, presents the data, method, and results.

2. NEW VOTING TECHNOLOGIES IN RUSSIA

New voting technologies come in two types: the tabulation of paper ballots by an electronic device, and electronic voting. In the former type, voters mark their preference on a paper ballot which is then tabulated automatically, while in the latter type, both the marking of the voter preference and the vote tabulation are done electronically [Alvarez and Hall 2010: 9]. Both types of new voting technologies have been used in Russia. On a very small scale, Russians in recent years have voted using electronic voting machines [*Kompleks Elektronnogo Golosovaniya – KEG*] that were designed and produced in Russia. Only 326 polling stations, or 0.3% of all polling stations, were equipped with such machines in the 2011 and 2012 elections. More widely used in Russia has been a type of 'optical scan voting system' or 'precinct count optical scanner' [IDEA 2011] called *Kompleks Obrabotki Izbiratel'nykh Byulletenei* [KOIB]. The KOIB was created in 2003, and has since been used in gradually increasing numbers: in the 2011 and 2012 national elections, roughly five percent of polling stations were equipped with KOIBs. Appendix 1 shows the distribution of KOIBs across Russia's 83 regions in these two elections.

The purpose of the KOIB, as of other optical scan voting systems, is to automatically read ballots and tally voting results. Once the results have been tallied, a printer that is connected to the KOIBs, which are always operated in pairs, prints a results protocol, while the results are also stored on a memory card inside the KOIB. Both the paper results protocol and the memory card are then transferred to the higher-level election commission, where the results are entered into the automated data-processing system which aggregates the results from all polling stations in the relevant election. The KOIBs make two types of voting fraud more complicated, if not impossible, to perpetrate – ballot-stuffing and tampering with the result protocol by members of election commissions [Lyubarev et al. 2007; Goldsmith 2011: 6]. Ballot-stuffing becomes difficult because ballots can be entered into the KOIB only one by one, and because they can be entered only slowly as the ballots are immediately scanned by the machine. Tampering with the results protocol is in principle impossible where KOIBs are used because the protocols are automatically produced by the KOIB system.

Russian authorities have often commended the introduction of new technology, including KOIBs, into Russian elections. In an article published in 2012, head of the Central Election Commission Vladimir Churov praised the KOIBs for being a domestically developed and produced technology, and for being more secure than some equivalents used in Western countries [Churov 2012]. Two years before, then prime minister Vladimir Putin ordered that by 2015 95% of voters use KOIBs in elections [Kornya 2013]. And Moscow mayor Sobyanin insisted on the use of a large number of KOIBs in the 2013 mayoral elections in Moscow, arguing that the KOIBs would ensure a fair election [Demidyuk 2013]. Despite these endorsements, it was announced in July 2013 that KOIBs, along with KEGs, will be phased out. The official reason for the abandonment of KOIBs is that, due to the increase in the number of political parties in Russia, the ballots that will be used in future elections will be longer than can be processed by the KOIBs [Subbotina 2013].

3. FRAUD IN THE 2011-2012 ELECTION CYCLE

Russians went to the polls in December 2011 to elect a new legislature, and again three months later to elect a new president. The legislative election was won by the ruling United Russia party, which is affiliated with Putin, with 49% of the vote, down from 64% four years earlier. Turnout in these elections was reported at 60%. A widespread perception of election fraud was one of the reasons behind the biggest street demonstrations in Russia since the early 1990s, that followed the election. The presidential election in March 2012 was won by Vladimir Putin, who previously served as president between 2000 and 2008. According to the official election results, Putin took home 64% of the vote, and turnout stood at 65%.

Electoral fraud is understood here as deliberate and illegal acts that are meant to distort electoral outcomes [Lehoucq 2003: 233; Vickery and Shein 2012: 9]. Since the elections, a flurry of research has produced insights into the scope, geographical distribution, and nature of fraud committed in the elections. Much of this research is 'election forensics', drawing from precinct-level data published on the website of the Central Election Commission of Russia and subordinate election commissions. Election forensics applied to the 2011 and 2012 election has drawn on previous research that especially found anomalous turnout patterns in the 2003-2004 and 2007-2008 election cycles [Mebane and Kalinin 2009; Myagkov et al. 2009]. Next to anomalies in turnout distribution, election forensics studies have also focused on irregular patterns in the distribution of second-digits and last digits in electoral returns that could point to manipulation [Beber and Scacco 2012; Deckert et al. 2011; Mebane 2011; Sjoberg 2012]. In the context of Russian elections, Mebane and Kalinin [2010] in particular revealed changes in the location of fraud in the 2007 and 2008 election by using second-digit mean tests. Election forensics analyses of the 2011 and 2012 elections reveal a high degree of anomalous turnout, not just in regions that are notorious for reporting incredible election results [such as Bashkortostan, Chechnya, and Mordovia], but across Russia. Turnout is often not only remarkably high, but it is disproportionately often also suspiciously approximate to round numbers: when the distribution of turnout in the 2011 and 2012 elections is plotted, spikes are visible at 70%, 75%, et cetera [Gehlbach 2012; GOLOS 2012b: 131; Shpil'kin 2012]. In the 2011 elections, for almost every polling station with anomalously high turnout, the party that benefits most from the surplus of votes is United Russia: whereas for all other parties turnout shows no discernible correlation with vote share, the vote share of United Russia is higher as turnout is higher [Golos 2012a: 237; Klimek et al.: 2012]. The same phenomenon is visible in the 2012 election with regard

to the Putin vote [Shpil'kin 2012]. The implication is that both reported turnout and the reported result for United Russia and Putin are associated with the incidence of fraud; the higher turnout and the result for United Russia and Putin are, especially relative to polling stations in the same administrative unit, the more likely it is that fraud has been committed to achieve the high turnout and the strong result for the ruling party and Putin [GOLOS 2012a: 232-238; GOLOS 2012b: 130-135; Shpil'kin 2011a; Shpilkin 2012c]. Turnout and vote share anomalies in the official results were more prevalent in the 2011 election than in the 2012 election. The research accordingly finds that fraud was committed on a bigger scale in 2011 than in 2012 [Kobak 2012; Shen' 2012].

A rich source of qualitative information about fraud in the 2011 and 2012 elections is the website www.kartanarusheniy.org [Map of Violation], where, through crowd-sourcing, over 13000 reports on irregularities in the two election were brought together. A study of the reports has found that the most common forms of fraud during the voting process, that were reported to the site, were ballot-stuffing, group voting, multiple voting, and vote-buying. Forms of fraud most often reported with regard to the vote count include intentional miscounting and tampering with the results protocol [Bader 2013]. As noted, two of these forms of fraud – ballot-stuffing and tampering with the results protocol – are made more difficult when KOIBs are used.

Since both ballot-stuffing and tampering with result protocols were widespread in the elections, we would expect polling stations with KOIBs to have lower turnout and a lower vote share for United Russia [in 2011] and Putin [in 2012] than polling stations without KOIBs, all else being equal. Russian bloggers have written about districts and cities where polling stations with KOIBs indeed had lower turnout and vote share than polling stations without KOIBs. In a number of districts in Mordovia, for example, polling stations with KOIBs on average reported 69% vote share for United Russia, while polling stations in the same districts without KOIB reported an average vote share of 89% [Shpil'kin 2012a]. Similarly, in the Yurginskiy district of Tyumen region, the two polling stations with KOIBs reported a vote share for United Russia in the low sixties, while all other polling stations, without KOIBs, had over ninety percent vote share for the ruling party [Kireev 2011]. These small-scale analyses directly compare results from polling stations with KOIBs, with results from polling stations without KOIBs in districts where some of the polling stations were equipped with KOIBs. For a larger study of the effect of the use of KOIBs in polling stations, however, this approach is inadequate because the polling stations with KOIBs are not randomly selected. The next section proposes and applies a simple, alternative method to assess the effect from the use of KOIBs.

4. DATA, METHODOLOGY, RESULTS

KOIBs were present in 76 out of the 83 regions of the Russian Federation in both the 2011 and 2012 elections [see appendix 1]. The remaining regions had small numbers of KEGs instead. In 2011, KOIBs were present in 4,828 of the roughly 95,000 polling stations in the country, against 5,249 in 2012. In neither election did the Central Election Commission of Russia reveal which polling stations were given KOIBs. Many regional election commissions, however, did provide information on the location of KOIBs inside the respective regions, suggesting that decisions about where to install the KOIBs were made at the regional level. One Russian blogger in particular has collected these regional-level data on the location of KOIBs for both elections [Shpil'kin 2011b; Shpil'kin 2012b], The data for some regions remain missing. Altogether, for 67 out of 76 regions with KOIBs do we know for both elections which polling stations had KOIBs. 4,063 KOIBs were present in these regions in 2011, against 4,447 in 2012. In 26 of the 67 regions, KOIBs were present in exactly the same polling stations in 2012 as in 2011. In six regions, by contrast, the location of all polling stations was different compared with the other election. There were 996 polling stations with KOIBs in the 2011 election that subsequently did not have KOIBs in the 2012 election. Conversely, there were 1,392 polling stations with KOIBs in the 2012 election that did not have KOIBs in the 2011 election.

In Russia and elsewhere, the borders of precincts and the numbering of polling stations often change from one election to the next. Due to the fact that the 2011 and 2012 elections were only three months apart, this happened in relatively few cases between these two elections, which makes it possible to directly compare most of the individual polling stations across the two elections. Of

the 996 polling stations that did have KOIBs in 2011 but not in 2012, 804 polling stations, divided over 111 districts, had the same number in 2012, or the corresponding 2012 polling station could easily be identified. Of the 1,392 polling stations that did have KOIBs in 2012 but not in 2011, the corresponding polling station in 2011 could be identified in 916 cases, divided over 110 districts. Altogether, then, the analysis is based on 1,720 [804 + 916] cases. Appendix 2 lists the number of cases per region.

Our method is based on analysis of the 'flow of votes' between elections, and follows a differences-in-differences design.[1] We compare the relative election returns – relative to the returns of polling stations in the same district – across the 2011 and 2012 elections of polling stations that did have KOIBs in 2011 but not in 2012, and of polling stations that did have KOIBs in 2012 but not in 2011. The design makes a parallel trend assumption, i.e. average change in turnout and incumbent vote share of polling stations with KOIBs in 2011 or 2012 is similar to that of polling stations in the same districts without KOIBs in both 2011 and 2012. The assumption is considered valid given that the districts are relatively small - comprising, on average, 37 polling stations - and geographically homogenuous. Take the following example. Polling station 1737 in the Leninsky district of Novosibirsk did use KOIBs in the 2011 election, but not in 2012. In the 2011 election, the polling station reported turnout of 56% and a 25% vote share for the ruling United Russia party. For the polling stations in the district that did not use KOIBs, mean turnout and United Russia vote share comprised 64% and 42%, respectively. For polling station 1737, consequently, turnout was 8% lower, and United Russia votes share 17% lower, than for polling stations without KOIBs in the district. The same polling station 1737 reported 65% turnout and 51% vote share for Putin in the 2012 election, while mean turnout and Putin vote share for the polling stations that did not have KOIBs in the 2011 elections, was 66% and 62% respectively. In 2012, then, the difference in turnout and vote share between polling station 1737, this time without KOIBs, and the polling stations that did not use KOIBs in 2011, was only 1% and 11% respectively, compared with 8% lower turnout and 17% lower vote share in 2011. By extension, the effect of the use of KOIBs [the 'KOIB effect'] in polling station 1737 in 2011 was a 7% decrease in turnout, and a 6% decrease in vote share for the incumbent candidate.

This calculation is performed for all 1,720 cases. For polling stations that did have KOIBs in 2011 but not in 2012, we find that the KOIB effect amounts to a decrease of 3.8% in turnout and a 4.8% decrease in vote share for the incumbent candidate. For polling stations that did have KOIBs in 2012 but not in 2011, the effect is much smaller: a 0.4% decrease in turnout, and a 0.6% decrease in incumbent vote share. Table 1 summarizes our findings. The absence of a considerable KOIB effect in the 2012 election can be explained in two ways. First, there was no fraud, or less fraud, in the districts with KOIBs in the 2012 election, that KOIBs could have prevented or reduced. Second, the forms of fraud that KOIBs are thought to forestall were prevalent despite the use of KOIBs. Considering the presence of a significant KOIB effect in 2011, however, the much smaller KOIB effect in 2012 seems to reflect the finding from election forensics analyses that the 2012 election was substantially less fraudulent than the 2011 election.

Table 1. The KOIB effect in the 2011 and 2012 elections

	number of polling stations	**KOIB effect on turnout**	**95% confidence interval**	**σ**	**KOIB effect on incumbent vote share**	**95% confidence interval**	**σ**
2011 legislative election	804	-3.8%	-4.4% to -3.2%	9.5%	-4.8%	-5.4% to -4.2%	8.4%
2012 presidential election	916	-0.4%	-0.9% to 0.1%	7.6%	-0.6%	-1.3% to 0.1%	11.1%

The KOIB effect was significant in the 2011 election. One possible explanation for the fact that polling stations reported lower relative turnout in 2011, when they were equipped with KOIBs, than in 2012, when they were not equipped with KOIBs, is that the KOIBs intimidate voters, or that voting with KOIBs produces more invalid ballots. For voters, however, the act of voting in a polling

[1] For examples of studies that equally look at the 'flow of votes', see Levin et al, 2009, and Myagkov et al., 2009

station with KOIBs is identical to the act of voting in a polling station without KOIBs. Besides, if the alternative explanation would be correct, the KOIB effect should also be visible in the 2012 elections. If we zoom in on the data, we observe a divide between regions with a highly significant KOIB effect, and regions seemingly without such an effect. To illustrate this, table 2 shows the KOIB effect in all regions with more than twenty cases of polling stations with KOIBs in one of the two elections but not in the other. A strong KOIB effect can be noted in the Chelyabinsk, Nizhnii Novgorod, and Novosibirsk regions. In other regions, the effect is mixed or rather insignificant. The relative lack of a KOIB effect in some regions can be related to a lower incidence of election fraud. Further research should shed light on the reasons behind the regional differences.

Table 2. The KOIB effect in selected regions

	KOIB effect on turnout 2011	**KOIB effect on incumbent vote share 2011**
Altay krai	-2.4%	-0.2%
Belgorod region	-0.7%	-0.9%
Chelyabinsk region	-12.8%	-22.1%
Chuvashia republic	-1.1%	-2.5%
Ivanovo region	1.4%	2.5%
Krasnoyarsk krai	1.5%	-0.6%
Nizhnii Novgorod region	-10.6%	-6.7%
Novosibirsk region	-6.6%	-6.7%
Perm krai	3.0%	-4.3%
Yaroslavl region	3.4%	-9.5%

We also find confirmation that the distribution of KOIBs at the district level was not random: polling stations that used KOIBs in the 2011 election but not in the 2012 election still had 2.3% lower turnout and 2.8% lower incumbent vote share on average than polling stations of the same district in 2012, when they did not use KOIBs. The equivalent figures for the 2012 elections are smaller: 0.3% and 1.3% respectively. It appears that, especially for the 2011 election, regional authorities disproportionately selected polling stations for the use of KOIBs that would also have had lower turnout and incumbent vote share when KOIBs would not have been used.

5. CONCLUSIONS

When new voting technologies are introduced, there are often legitimate concerns that these technologies can be manipulated to commit electoral fraud. It stands to reason that such concerns particularly arise in the context of an electoral authoritarian regime such as Russia's. Rather than being conducive to more fraud, however, this study indicates that the optical scan voting systems that have been used in Russia over the past decade reduce rather than enhance the incidence of fraud. The likely explanation is that, by automating the vote count, the optical scan voting systems take human involvement out of part of the voting and counting process and thus narrow the scope for fraud.

The finding that new voting technologies can contribute to reducing the incidence of electoral fraud in an undemocratic state has implications for international and non-governmental organizations as well as governments that promote democratic elections. From the perspective of these stakeholders, the decision by the Russian authorities to abandon new voting technologies should be a matter of concern. While the introduction of new voting technologies always warrants close scrutiny, the findings from this study suggest that there may be sense in promoting their wider application in states with a known record of electoral fraud.

APPENDIX 1. KOIBS IN THE 2011 AND 2012 ELECTIONS

Region	2011 Duma election	2012 Presidential election
Adygeya republic	14	14
Altay republic	13	13
Altay krai	100	100
Amur region	40	40
Arhangelsk region	53	53
Astrakhan region	35	35
Bashkortostan republic	178	178
Belgorod region	61	61
Bryansk region	58	58
Buryatiya republic	41	41
Chechnya republic	0	0
Chelyabinsk region	111	111
Chukotka autonomous region	3	3
Chuvashia republic	59	59
Dagestan republic	92	92
Ingushetiya republic	7	8
Irkutsk region	94	94
Ivanovo region	39	40
Jewish autonomous region	8	8
Kabardino-Balkaria republic	0	0
Kaliningrad region	27	27
Kalmykia republic	14	15
Kaluga region	37	37
Kamchatka krai	17	18
Karachaevo-Cherkessia republic	13	13
Karelia republic	30	30
Kemerovo region	87	87
Khabarovsk krai	42	43
Khakhasia republic	0	0
Khanty-Mansiyskiy autonomous region	30	30
Kirov region	66	66
Komi republic	0	0
Kostroma region	35	35
Krasnodar krai	132	132
Krasnoyarsk krai	109	112
Kurgan region	59	59
Kursk region	61	61
Leningrad region	49	49
Lipetsk region	46	46
Magadan region	7	7
Mariy El republic	0	0
Mordovia republic	44	45
Moscow	250	250
Moscow region	167	167
Murmansk region	0	0
Nenetskiy autonomous region	3	3
Nizhny Novgorod region	123	119
North Ossetia	19	19
Novgorod region	28	28
Novosibirsk region	101	101
Omsk region	95	95
Orenburg region	92	92
Oryol region	40	40
Penza region	59	59
Perm krai	95	95
Primorye krai	77	77
Pskov region	33	33
Rostov region	132	132
Ryazan region	55	55
Saint-Petersburg	20	120
Sakha [Yakutia]	41	41
Sakhalin region	23	23
Samara region	88	88
Saratov region	92	92
Smolensk region	41	41
Stavropol krai	65	65
Sverdlovsk region	300	600
Tambov region	52	52
Tatarstan republic	0	0
Tomsk region	100	110
Tula region	56	61
Tver region	67	67
Tyumen region	58	58
Tyva republic	10	10
Udmurtiya republic	59	59
Ulyanovsk region	51	51
Vladimir region	49	49
Volgograd region	85	85
Vologda region	100	100
Voronezh region	84	84
Yamalo-Nenetskiy autonomous region	11	11
Yaroslavl region	47	47
Zabaykalskiy krai	49	50
total	4828	5249

APPENDIX 2. NUMBER OF CASES

Region	2011 Duma election	2012 Presidential election
Adygeya republic	0	0
Altay republic	0	0
Altay krai	54	62
Amur region	0	0
Arhangelsk region	4	5
Astrakhan region	0	0
Bashkortostan republic	0	0
Belgorod region	27	27
Bryansk region	0	0
Buryatiya republic	0	0
Chechnya republic	0	0
Chelyabinsk region	43	43
Chukotka autonomous region	0	0
Chuvashia republic	61	59
Dagestan republic	17	18
Ingushetiya republic	0	1
Irkutsk region	0	0
Ivanovo region	39	40
Jewish autonomous region	0	0
Kabardino-Balkaria republic	0	0
Kaliningrad region	8	7
Kalmykia republic	3	4
Kaluga region	0	0
Kamchatka krai	1	2
Karachaevo-Cherkessia republic	9	9
Karelia republic	0	0
Kemerovo region	5	5
Khabarovsk krai	0	0
Khakhasia republic	0	0
Khanty-Mansiyskiy autonomous region	0	0
Kirov region	0	0
Komi republic	0	0
Kostroma region	0	0
Krasnodar krai	18	17
Krasnoyarsk krai	54	54
Kurgan region	0	0
Kursk region	10	10
Leningrad region	0	0
Lipetsk region	4	4
Magadan region	0	0
Mariy El republic	0	0
Mordovia republic	0	0
Moscow	3	0
Moscow region	0	0
Murmansk region	0	0
Nenetskiy autonomous region	0	0
Nizhny Novgorod region	123	119
North Ossetia	0	0
Novgorod region	0	0
Novosibirsk region	102	101
Omsk region	0	0
Orenburg region	0	0
Oryol region	0	0
Penza region	8	12
Perm krai	95	95
Primorye krai	5	4
Pskov region	8	8
Rostov region	0	0
Ryazan region	3	3
Saint-Petersburg	0	105
Sakha [Yakutia]	0	0
Sakhalin region	0	0
Samara region	0	0
Saratov region	0	0
Smolensk region	11	11
Stavropol krai	22	26
Sverdlovsk region	0	0
Tambov region	4	4
Tatarstan republic	0	0
Tomsk region	11	10
Tula region	0	7
Tver region	0	0
Tyumen region	0	0
Tyva republic	0	0
Udmurtiya republic	0	0
Ulyanovsk region	0	0
Vladimir region	0	0
Volgograd region	0	0
Vologda region	0	0
Voronezh region	0	0
Yamalo-Nenetskiy autonomous region	5	3
Yaroslavl region	47	41
Zabaykalskiy krai	0	0
total	804	916

REFERENCES

Alvarez, R. Michael, and Thad E. Hall. 2010. *Electronic elections: The perils and promises of digital democracy*. Princeton University Press.

Bader, Max. 2013. Crowdsourcing election monitoring in the 2011–2012 Russian elections. *East European Politics* 29[3].

Beber, Bernd, and Alexandra Scacco. 2012. What the Numbers Say: A Digit-Based Test for Election Fraud. *Political Analysis* 20[2]: 211-234.

Buzin, Andrei. 2013. "Lebedinaya pesnya moskovskikh KOIBov" Available at: http://abuzin.livejournal.com/123575.html

Churov, Vladimir E. 2012. Tekhnicheskoe obespechenie vyborov i elektronnogo golosovaniya v Rossiiskoi Federatsii. Available at: http://www.cikrf.ru/news/relevant/2012/07/19/doclad_churov_tech.html

Deckert, Joseph, Mikhail Myagkov, and Peter C. Ordeshook. 2011. Benford's Law and the Detection of Election Fraud. *Political Analysis* 19[3]: 245-268

Demidyuk, Natal'ya. 2013. *Sobyanin vystupil za izpol'zovanie KOIBov*. Available at: http://www.mk.ru/moscow/article/2013/07/19/886738-sobyanin-vyistupil-za-ispolzovanie-koibov.html

Gehlbach, Scott. 2012. *Electoral Fraud in Russia. Report from the Russian Blogosphere*. Available at. http://themonkeycage.org/blog/2012/01/27/electoral-fraud-in-russia-report-from-the-russian-blogosphere/

Goldsmith, Ben. 2011. *Electronic Voting & Counting Technologies: A Guide to Conducting Feasibility Studies*. International Foundation for Electoral Systems

Golos. 2012a. *Vybory v Rossii 4 dekabrya 2011 goda. Analiticheskii Doklad*. Available at. http://www.golos.org/news/4567

Golos. 2012b. *Vybory Prezidenta Rossii 4 Marta 2012 goda. Analiticheskii Doklad*. Available at http://files.golos.org/docs/6088/original/6088-doklad-2012-03-04-ok3.pdf?1338893979

IDEA. 2011. *Introducing Electronic Voting. Essential Considerations*. Policy Paper, December 2011.

Klimek, Peter, Yuri Yegorov, Rudolf Hanel, and Stefan Thurner. 2013. *It's not the voting that's democracy, it's the counting: Statistical detection of systematic election irregularities*. Available at http://arxiv.org/pdf/1201.3087v3.pdf

Kireev, Aleksandr. 2011. "Na kakikh dvukh uchastkakh byli ustanovleny KOIBy?" Available at: http://kireev.livejournal.com/707980.html

Kobak, Dmitry. 2011. "PZhiV i, proshu proshcheniya, koiby". Available at: http://kobak.livejournal.com/103331.html

Kobak, Dmitry. 2012. "O rezul'tatakh vyborov, po-bystromu". Available at: http://kobak.livejournal.com/104514.html

Kornya, Anastasiya. 2013. "Elektronnogo podshchyota na vyborakh ne budet" *Vedomosti* 9 July.

Lehoucq, Fabrice E. 2003. Electoral Fraud: Causes, Types, and Consequences. *Annual Review of Political Science* 6: 233-256.

Levin, Inés, Gabe A. Cohn, Peter C. Ordeshook, and. R. Michael Alvarez. 2009. *Detecting Voter Fraud in an Electronic Voting Context: An Analysis of the Unlimited Reelection Vote in Venezuela*. Available at: https://www.usenix.org/legacy/event/evtwote09/tech/full_papers/levin.pdf

Loeber, Leontine. 2008. "E-Voting in the Netherlands; from General Acceptance to General Doubt in Two Years." Proceedings of the 3rd international Conference on Electronic Voting 2008, August 6th - 9th , 2008, Bregenz.

Lyubarev, A.E., Buzin, A.Yu, and Kynev, A.V. 2007. *Mertvye Dushi: Metody fal'sifikatsii itogov golosovaniya i bor'ba s nimi*. Moscow: Nikkolo M.

Mebane, Walter R. 2011. Comment on "Benford's Law and the Detection of Election Fraud" *Political Analysis* 19[3]: 269-272

Mebane Jr., Walter, and Kirill Kalinin. 2009. Elektoral'nye fal'sifikatsii v Rossii: kompleksnaya diagnostika vyborov 2003-2004, 2007-2008 gg. *Rossiyskoe Elektoral'noe Obozrenie* 2: 57-70.

Mebane Jr., Walter, and Kirill Kalinin. 2010. Electoral Fraud in Russia: Vote Counts Analysis using Second-digit Mean Tests. Prepared for presentation at the Annual Meeting of the Midwest Political Science Association, Chicago, IL, April 22-25, 2010

Myagkov, Mikhail G., Peter C. Ordeshook, and Dimitri Shakin. 2009. *The Forensics of Election Fraud: Russia and Ukraine*. Cambridge & New York: Cambridge University Press.

Pshenichnikov, Maksim. 2011. "Otgadko". Available at: http://oude-rus.livejournal.com/545739.html

Sergeeva, Viktoria. 2013. "Sergeiu Sobyaninu ponadobilis' elektronnye urny". *Kommersant* 19 July.

Shen', A. Vybory i statistika: kazus Edinoy Rossii [2009-2012]. Available at: http://arxiv.org/pdf/1204.0307.pdf

Shpil'kin, Sergei. 2011a. *Statistika issledovala vybory*. Available at: Gazeta.ru. http://www.gazeta.ru/science/2011/12/10_a_3922390.shtml.

Shpil'kin, Sergei. 2011b. "Tekhnicheskiy post – spisok uchastkov s KOIB". Available at: http://podmoskovnik.livejournal.com/125484.html

Shpil'kin, Sergei. 2012a. "KOIBy v Mordovii". Available at: http://podmoskovnik.livejournal.com/137809.html

Shpil'kin, Sergei. 2012b "Spisok uchastkov s KOIB/KEG na presidentskikh vyborakh". Available at: http://podmoskovnik.livejournal.com/141910.html

Shpil'kin, Sergei. 2012c. *"Yavka opyat' srabotala v pol'zu odnogo iz kandidatov"* Available at. Gazeta.ru. http://www.gazeta.ru/science/2012/03/06_a_4026805.shtml.

Sjoberg, Fredrik M. 2012. Making Voters Count: Evidence from Field Experiments about the Efficacy of Domestic Election Observation. Harriman Institute Working Paper No. 1. Available at: http://harriman.columbia.edu/files/harriman/newsletter/Harriman%20Working%20Paper%201_0.pdf

Subbotina, Svetlana. 2013. "Yedinorossy reshili okonchatel'no zapretit' KOIBy". *Izvestiya* 13 July.

Vickery, Chad, and Erica Shein. *Assessing Electoral Fraud in New Democracies: Refining the Vocabulary*. Washington, D.C.: International Foundation for Electoral Systems.

Volkamer, Melanie. 2010. "Electronic Voting in Germany." In Sergei Gutwirth, Yves Poullet, Paul de Hert [eds.] *Data Protection in a Profiled World*. Springer Netherlands, 2010. 177-189.

USENIX Journal of Election Technology and Systems (JETS)
Volume 2, Number 1 • December 2013
www.usenix.org/jets/issues/0201

Faster Print on Demand for Prêt à Voter

CHRIS CULNANE, University of Surrey
JAMES HEATHER, University of Surrey
RUI JOAQUIM, University of Luxembourg
PETER Y. A. RYAN, University of Luxembourg
STEVE SCHNEIDER, University of Surrey
VANESSA TEAGUE, The University of Melbourne

Printing Prêt à Voter ballots on demand is desirable both for convenience and security. It allows a polling station to serve numerous different ballots, and it avoids many problems associated with the custody of the printouts. This paper describes a new proposal for printing Prêt à Voter ballots on demand. The emphasis is on computational efficiency suitable for real elections, and on very general ballot types.

1. INTRODUCTION

There are several projects on end-to-end verifiable election protocols either already used, or intended to be used in government elections, including the Scantegrity II project in Takoma Park [Carback et al. 2010], and the StarVote project in Texas [Benaloh et al. 2011]. Various other e2e systems such as VoteBox [Sandler et al. 2008], Wombat [Rosen et al. 2012] and Helios [Adida 2008] have been used for non-government elections. Many other proposals exist in the academic literature.

Our motivation is adapting Prêt à Voter for elections in the Australian state of Victoria for the 2014 state election. This requires STV[1] ballots with approximately 30 candidates, and various other ballot types. Although the proposed algorithm is specific to the Victorian project, it would work just as well for other versions of Prêt à Voter, including those that are filled in manually.

This paper introduces a new protocol for the verifiable printing on demand of Prêt à Voter ballot forms. A similar approach would work for any scheme in which voters select or arrange ciphers from a pre-generated ballot form. Scantegrity is the obvious other application. The crucial requirement is that voters (and others) can audit some ballot forms for correctness *without compromising voter privacy*, because the audit occurs before the person votes. Voters then vote on ballot forms that have not been audited.

The integrity of Prêt à Voter depends crucially on proper construction of the printed ballot forms, meaning that the plaintext candidate list that the voter sees must match the encrypted values for that ballot. Consequently this proposal details opportunities for auditing their correct construction and printing. The proposed construction is very computationally efficient and retains most of the desirable properties of existing print-on-demand proposals in the literature. The information flow of our scheme is similar to Markpledge 2 [Adida and Neff 2009], though the auditing is different. The main idea is that the device encrypts the vote directly using randomness generated by others.

Also the system must address the question of "kleptographic" privacy attacks [Gogolewski et al 2006], in which the (public) ciphertexts contain deliberately poorly-chosen randomness that exposes the vote. This is possible whenever the entity building the ciphertexts also controls all the randomness used. This problem is addressed by distributing the process of inputting randomness into ciphertexts.

In summary, distributed ballot generation is a desirable property to have for the following reasons:

— ensuring the candidate lists are randomly generated,
— ensuring no single generating entity knows all the (plaintext) candidate lists, and

[1] Single Transferable Vote (STV) is a proportional representation system that allows voters to number many candidates in their order of preference.

This work is supported by the UK Engineering and Physical Sciences Research Council, under grant EP/G025797/1, and by Fonds National de la Recherche Luxembourg, under the SeRTVS project.

— ensuring extra information about the candidate list can't be leaked in the ballot ciphertexts (as in kleptographic attacks).

This proposal is designed so that the entity that builds the ciphertexts (the printer) has a deterministic algorithm to follow. This provides a way to distribute the expensive cryptographic operations to the network of printers, whilst retaining the central, distributed, generation of randomness to maintain the three properties above.

1.1. Protocol overview

Our protocol has two roles. The "randomness generation servers," of which a threshold are trusted for privacy, send randomness to a "printer". The "printer" uses only that randomness to generate the ballots, which it can then print on demand. In brief:

(1) Each randomness generation server generates some randomness, commits to it publicly, and sends the opening secretly to the printer.
(2) The printer uses that randomness to generate the encrypted ballot, which it publishes.
(3) When required, the printer prints the next ballot in sequence, with human-readable candidate names.

There are thus two important points for public auditing:

(1) An audit of the encrypted ballot produced in step 2, to check that the candidate ciphertexts are valid and that the printer used the proper randomness. This is described in Section 2.5.
(2) A standard Prêt à Voter audit of the printed ballot from step 3, to check that the printed human-readable candidate names match those of the encrypted ballot. This is described for our scheme in Section 3.2.

1.2. Prior work and our contribution

There are several papers that provide constructions for distributed ballot generation in Prêt à Voter [Ryan and Schneider 2006; Ryan 2007; Ramchen and Teague 2010]. However, most of these do not extend to IRV/STV ballots, and those that do require a substantial amount of cryptographic computation. Our motivating application for Victorian state elections requires approximately 60 ciphertexts to be generated, permuted, and committed for each ballot.[2] These ciphertexts must also be securely obtained by printers in polling stations. The computational cost of generating a sufficiently large number of ballots using existing methods (such as [Ramchen and Teague 2010]), is too high.

In our scheme, deviations from proper ballot generation will be detected with bounded probability by audit. This is weaker than the pure cryptographic approaches, which involve a zero knowledge proof from which deviations are detected with overwhelming probability. Our approach allows auditing to make the error bound arbitrarily small, but it could require a lot of auditing. This seems a reasonable tradeoff given the computational efficiency gain, since improper ballot *printing* must be detected by random audits anyway. See Section 6.4 for more discussion of this issue.

Although this approach would work for simpler ballot types, the tradeoff is only justifiable if the cryptographic approaches that have an overwhelming probability of detecting cheating are infeasible. For first past the post and other simple voting methods, existing protocols in the literature would be quite feasible.

It seems likely that our techniques would extend to schemes such as Wombat, StarVote, VoteBox and Helios, in which the device encrypts the vote directly. The idea would be to extend the "challenge" process available in those systems to include a check that the device had used the right randomness. However, without more careful analysis of each particular scheme it is difficult to be certain that this would completely remove all opportunities for kleptographic attacks. Nor would this proposal necessarily represent the right tradeoff between efficiency and security for those schemes.

[2]This includes the 30-candidate STV list and two other voting methods each involving approximately 15 candidates.

Our approach has similar aims to Backes *et al.*'s privacy preserving accountable computation [Backes et al. 2013] and to Verifiable Random Functions [Micali et al. 1999]. However, those works have a single trusted generator of the randomness, which we need to avoid, though in both cases it might be possible to generate the secret information in a distributed way and then publish the public information. Also, both constructions requires a zero knowledge proof to be generated for every computation, which is not efficient enough for our application.

Throughout this document when we refer to the "Printer" we are in fact referring to the tablet device that is connected to the printer. As such, the "Printer" has the processing power you would expect to find on a mid-range tablet. The "Web Bulletin Board (WBB)" is an authenticated broadcast channel with memory. It also returns a signature on messages posted on it, and performs some basic tests for correctness before accepting posts.

The next section describes how ballots are generated and how the generation is audited. Section 3 describes how they are printed on demand and how printing is audited. Section 4 descibes an optimisation. Section 5 describes some practical details related to runtime failures. Section 6 gives an informal security analysis.

2. BALLOT GENERATION

2.1. Overview

The main idea is that the printer generates a permuted list of candidate ciphers using randomness values generated by a distributed set of peers. As such the printer undertakes the expensive crypto operations, but does not have any influence over the values used in those operations. This prevents the printer from mounting kleptographic attacks or otherwise having any influence over the ciphertexts.

During ballot generation the printer is audited to ensure that it has performed honestly: if a sufficient number of ballots are audited and shown to be correct then we can gain a high assurance that the printer has behaved honestly. The definition of "honest" is different from standard versions of Prêt à Voter, in which a dishonest printer can only misalign the printed candidate names with the ballot onion. In our version, a dishonest printer may also attempt to generate invalid ballot ciphertexts or perform a kleptographic attack by using randomness other than that specified by the protocol. However, these two kinds of cheating can be detected by a ballot-generation audit—see Section 2.5.

Notation:

$Enc_k(m;r)$. is the encryption of message m with public key k and randomness r.[3]
$Dec_k(m)$. is a decryption of m using the private key k.
$ReEnc(\theta;r)$. is a re-encryption (re-randomisation) of the ciphertext θ using the randomness r (this abstracts the requirement of knowing the public key, which is a requirement for ElGamal re-encryption).
$c(m)$. is a perfectly hiding commitment to message m (using some randomness not explicitly given).
$c(m;r)$. is a perfectly hiding commitment to message m (using randomness r).[4]
PK_E. is the election public key (which is thresholded).
PK_P. is the printer's public key (which is not thresholded/distributed).
SK_P. is the printer's private key.
n. is the number of candidates.
b. is the number of ballots to be generated for each printer.
G. is the number of randomness generation servers.
$RGen_i$. is the i-th randomness generation server.
$SymmEnc_{sk}(m)$. is a symmetric-key encryption of m under the symmetric key sk.

[3]The Victorian project uses Elliptic Curve El Gamal.
[4]In the vVote project we use the hash-based commitment scheme described in [Jakobsson et al. 2002].

Candidate Name	ID
Vladimir Putin	$cand_1$
Mohamed Morsi	$cand_2$
...	...
Mahmoud Ahmadinejad	$cand_n$

Fig. 1. Initial Ballot Input: Candidate Identifiers

PrinterA:1	...	PrinterB:1	...	PrinterC:1
PrinterA:2	...	PrinterB:2	...	PrinterC:2
⋮	⋮	⋮	⋮	⋮
PrinterA:b				

Fig. 2. Initial Ballot input: Serial numbers for printers A,B,C.

$h(m)$. is a cryptographic hash of the message m.[5]
$SymmDec_{sk}(m)$. is a symmetric-key decryption of m using key sk.

We will post on the public WBB values that are encrypted with a threshold key, or perfectly hiding commitments. We will not post values that are encrypted with a non-thresholded key. We could have used computationally hiding commitments or encryptions with non-thresholded keys, but either of these would have meant that a single leak of relevant parameters, even quite a long time in the future, could have been combined with WBB data to violate ballot privacy. Our system does not achieve everlasting privacy, but it achieves a somewhat related weaker property, that no single entity's data (apart from the printer's) is enough to break ballot privacy, even given WBB data.

2.2. Pre-Ballot Generation

Before the ballot generation starts the following must occur:

(1) The election public key sharers jointly run a distributed key generation protocol to generate a thresholded private key and joint public key PK_E.[6]
(2) A list of candidate identifiers is generated and posted on the public WBB, as shown in Figure 1. Candidate identifiers are arbitrary, distinct elements of the message space of the encryption function.[7]
(3) For each printer, a list of serial numbers of the form "PrinterID:index" is deterministically generated and posted on the public WBB. This serial number is just the literal string as given. These serve as row indices for later computation.
(4) Each printer constructs a key pair and publishes the public key PK_P.

All this data, which is posted on the WBB immediately before ballot generation, is shown in Figures 1 and 2.

The following sections describe the process of ballot generation for a single printer, but it should be clear how the same process will be run in parallel for each printer.

[5]We use 256-bit AES and SHA-256 respectively. This means that the computational difficulty of guessing a key (2^{256}) is much greater than that of finding a collision (2^{128}). However, this seems justifiable since collision-finding is only useful for cheating during ballot generation, which must be performed in a restricted time, while guessing the symmetric key can be used to break ballot privacy long after the election.
[6]We keep the key sharers and the randomness generation servers conceptually separate, even if we end up using the same servers.
[7]For vVote, candidate identifiers are elliptic curve points either randomly selected or calculated as part of the optimisation used to speed up mixing, depending on the type of race.

Serial No	Encrypted Randomness		
PrinterA:1	$SymmEnc_{sk_i}(r_{1,1}\|\|R_{1,1})$	...	$SymmEnc_{sk_i}(r_{1,n}\|\|R_{1,n})$
PrinterA:2	$SymmEnc_{sk_i}(r_{2,1}\|\|R_{2,1})$	...	$SymmEnc_{sk_i}(r_{2,n}\|\|R_{2,n})$
⋮	⋮	⋮	⋮
PrinterA:b	$SymmEnc_{sk_i}(r_{b,1}\|\|R_{b,1})$	...	$SymmEnc_{sk_i}(r_{b,n}\|\|R_{b,n})$

Fig. 3. Ballot Input: Table RT_i, sent privately from peer i to printer A without public posting.

Serial No	Committed Randomness		
PrinterA:1	$c(r_{1,1};R_{1,1})$	...	$c(r_{1,n};R_{1,n})$
PrinterA:2	$c(r_{2,1};R_{2,1})$	...	$c(r_{2,n};R_{2,n})$
⋮	⋮	⋮	⋮
PrinterA:b	$c(r_{b,1};R_{b,1})$	...	$c(r_{b,n};R_{b,n})$

Fig. 4. Commitment to Initial Ballot Input: Table CRT_i, posted by peer i on the WBB.

2.3. Randomness Generation

The randomness generation consists of each server $RGen_i$ generating a large table of secret random values and sending them (privately) to the printer after posting (public) commitments to them on the WBB.

2.3.1. Notation

RT_i. is $RGen_i$'s private table of encrypted random values (Fig 3).
CRT_i. is $RGen_i$'s public table of commitments to the values in RT_i (Fig 4).

2.3.2. Detailed algorithm. In detail: each random value has length k, where k is a security parameter which should be about 256 bits. Each server $RGen_i$ generates a random symmetric key sk_i. It then generates a table of $b*n$ pairs of pieces of random data, each of size $2*k$ bits and encrypted under sk_i. The table is denoted by RT_i, and each pair can be retrieved by the serial number and column, or the row and column.[8] The result is shown in Figure 3. The idea is that the first element of each pair, $r_{(row,col)}$, will be used later by the printer; the second element, $R_{(row,col)}$, is used to commit to $r_{(row,col)}$ and to open the commitment in case of audits.

Each server commits to $r_{(row,col)}$ by posting on the WBB a commitment to it using randomness $R_{(row,col)}$. The table of commitments is shown in Figure 4. Call the table CRT_i. Each peer $RGen_i$ posts its CRT_i, and checks all CRT_i's are posted before sending RT_i privately to the printer.[9] $RGen_i$ also encrypts sk_i with the printer's public key and sends the result (denoted $esk_i = Enc_{PK_P}(sk_i;r)$) to the printer.

2.4. Ballot Permutation and Commitment

The printer receives the respective RT_i and corresponding (encrypted) key esk_i from each server. The printer also downloads or constructs the candidate identifiers and serial numbers shown in Figures 1 and 2. The printer now needs to encrypt and permute the candidate identifiers.

For each candidate identifier $cand_k$ in the pre-committed table in Figure 1 it encrypts it using the combined randomness from the RT tables, received from the randomness generation servers. To combine the randomness the printer first decrypts the encrypted symmetric key esk_i received from

[8] For example $RGen_i.RT_i(PrinterA:1,2)$ and $RGen_i.RT_i(1,2)$ will return the pair of encrypted random values in the second column for the first ballot: $SymmEnc_{sk_i}(r_{1,2}||R_{1,2})$.

[9] This is to stop the last peer choosing their randomness when they know the others'. If this was not enforced, then one bad randomness generation server colluding with a printer could determine the randomness values for each of that printer's ballots, thus breaking privacy. The bad server would wait until the printer told it all the other servers' random values, then generate its own to produce a particular final value.

each server and then uses the resulting key sk_i to decrypt the randomness in RT_i. For each element of each RT, the printer checks that the decrypted pair $r_{row,col}, R_{row,col}$ opens the commitment at $CRT_{row,col}$. It challenges any that do not—see Section 2.4.1 for this case.

The decrypted first elements $r_{row,col}$ from each peer are concatenated and hashed. The printer then uses the hash output for randomness when encrypting the candidate identifier under PK_E. The resulting ciphers are then sorted into canonical order to produce a random permutation. These ciphers are posted on the public WBB. Note that the output of the encryption is pseudo-random and as such sorting the encrypted ciphers will give a pseudo-random permutation π. The printer retains this permutation so that it can print the plaintexts in the appropriate order when requested to print that ballot. After the ciphers are submitted to the WBB, the audit protocol detailed in Section 3.2 can be run. The algorithm run by the printer is given in Algorithm 1.

Algorithm 1: Deterministic Encryption by Printer

for $i = 1 \to G$ **do** ▷ Decrypt the symmetric keys from the *RGen*Servers
  $sk_i \leftarrow Dec_{SK_P}(esk_i)$
end for
for $j = 1 \to b$ **do** ▷ b is the number of ballots.
  for $k = 1 \to n$ **do** ▷ n is the number of candidates.
    $rand \leftarrow SHA(SymmDec_{sk_1}(RT_1(j,k)) \| \ldots \| SymmDec_{sk_G}(RT_G(j,k)))$
    $CT_{j,k} \leftarrow Enc_{PK_E}(cand_k; rand)$
  end for
  $CT_{(j)} \leftarrow Sort(CT_{(j)})$ ▷ $CT_{(j)}$ returns the entire row
  $perms_j \leftarrow \pi$ ▷ π is the permutation applied to sort CT_j.
end for
Send CT to WBB.

The intention is that only the printer knows which ciphertexts correspond to which candidates, but its algorithm for generating those ciphertexts is deterministic. Hence it cannot use the ciphertexts to leak information without detection. Of course, the printer could always leak that information via a side channel, but this is unavoidable and occurs with any form of electronic ballot printing or marking.

2.4.1. What the printer should do if $RGen_i$ cheats. It's important in the above protocol that the printer checks the opening of each commitment, *i.e.* checks that for each element of each RT, the decrypted pair $r_{row,col}, R_{row,col}$ opens the commitment at $CRT_{row,col}$. It is important in practice that the printer raise an alarm on any commitments that are not correctly opened. Exactly how such a dispute should be resolved requires some careful engineering of procedures. It is difficult to tell whether the printer or the randomness generation server is misbehaving without exposing private ballot data. This is not necessarily a problem, because the randomness contributed by other randomness generation servers would not be exposed. Hence the other ballots remain private.

Note that the issue does not affect public verifiability, because the absence of proper commitment opening would be detected by an audit of this ballot. It does, however, affect accountability: if an audit detects that the value used to encrypt a ballot was not a valid opening of the commitment on the WBB, we would like to know whether it was the randomness generation server or the printer that cheated. If we insist that the printer performs this check, then we can be certain that a failed audit is the printer's fault.

2.4.2. Alternative construction with PRNGs. Rather than generate a separate random value for each candidate in each ballot, an alternative is to generate one random value for each ballot, then use a cryptographic PRNG to expand it to produce randomness for all of the encrypted values in the ballot. This introduces an assumption on the good expanding behaviour of the PRNG, but substantially reduces the communication costs of the protocol. Hence the tables of Figures 3 and 4 need only 2 columns rather than $n+1$. It does not significantly change auditing.

However, we have not taken this approach for the Victorian system due to practical concerns about PRNG implementations. The most important is the possibility that variations in implementations could imply a failure of reproducibility of the random sequences, which is required for audit. In principle this shouldn't be a difficult issue to address. In practice we were concerned this would make it significantly harder to write an independent verifier, so we opted to omit the PRNGs.

2.5. Ballot Generation Auditing

Someone chooses a suitable percentage of the ballots to audit at random.[10] For each ballot selected, the printer posts on the WBB the randomness it used during the generation, *i.e.* to open the commitments for that SerialNo in each peer's *CRT* table. The printer can either have this stored, or else can recalculate it from the encryptions it received prior to ballot generation. Anyone can verify the commitment openings $(r_{row,col}, R_{row,col})$ and reconstruct the ballot ciphertexts from them. Thus anyone can check that the ballots were correctly constructed and that the printer used the appropriate randomness.

3. PRINT ON DEMAND

This section describes what happens when a voter appears at a polling place. The printer needs to print a pre-generated ballot for the appropriate district. The printer knows the plaintexts and permutation for a particular ballot so can easily print the appropriate ballot out.

However, there is a risk that a misbehaving printer might print a completely invalid ballot, *i.e.* one that hasn't been part of the generation process described above. Although this is detectable by audit, we prefer for practical reasons to prevent it altogether. Hence we require that the printer obtain a signature from the WBB in order to create an authentic ballot. The WBB is attesting to those ciphertexts matching what the printer has already committed to.

The vVote project uses a combined EBM and scanner rather than the traditional Prêt à Voter technique of filling in the ballot with a pencil and then scanning it. The print on demand protocol works exactly the same either way, so the description below is intended to work for either the traditional pencil and scanner setup, or the vVote-style combined scanner and EBM.

Figure 5 shows the message sequence chart for print on demand, which is elucidated in Section 3.1 below. The authorisation, ballot reduction and serial number signing all take place together in a single round of communication. The signatures generated are deterministic BLS signatures [Boneh et al. 2004], including the signature from the WBB.

3.1. Ballot Reduction

It is unpredictable exactly how many of each ballot will be required at each location. We could generate an abundant oversupply of ballots with exactly the right number of candidates for each division, but this could be quite expensive. Instead it is probably more efficient to generate an abundant oversupply of generic ballots with a larger than necessary number of candidates, then reduce it down to the appropriately sized ballot for the district/region it is going to be used in. This allows great flexibility about who votes at what polling place—any voter can arrive anywhere and have a ballot produced to match their voting eligibility.

If the ballot contains more ciphers than candidates, we need to reduce the ballot in a manner that can be verified. The following proposal has the nice feature that the voter (or the EBM if there is one) does not need to know where the blanks are in order to cast the vote: they just get a candidate list in the order that the ciphertexts for the candidates they need appear on the ballot. Then the EBM or scanner prints, and the voter checks, the voter's preference numbers against that order.

Suppose from now on there are m candidates in the division and n $(> m)$ ciphertexts on the ballot. ($n = m$ is a special case of the steps below.) The printer is supposed to use the ciphertexts for $cand_1, cand_2, \ldots, cand_m$. Of course it could cheat and attempt to use other ciphertexts instead,

[10] The questions of who chooses, how they choose, and how it can be guaranteed that they choose well enough to engender confidence in a particular election result are beyond the scope of this paper.

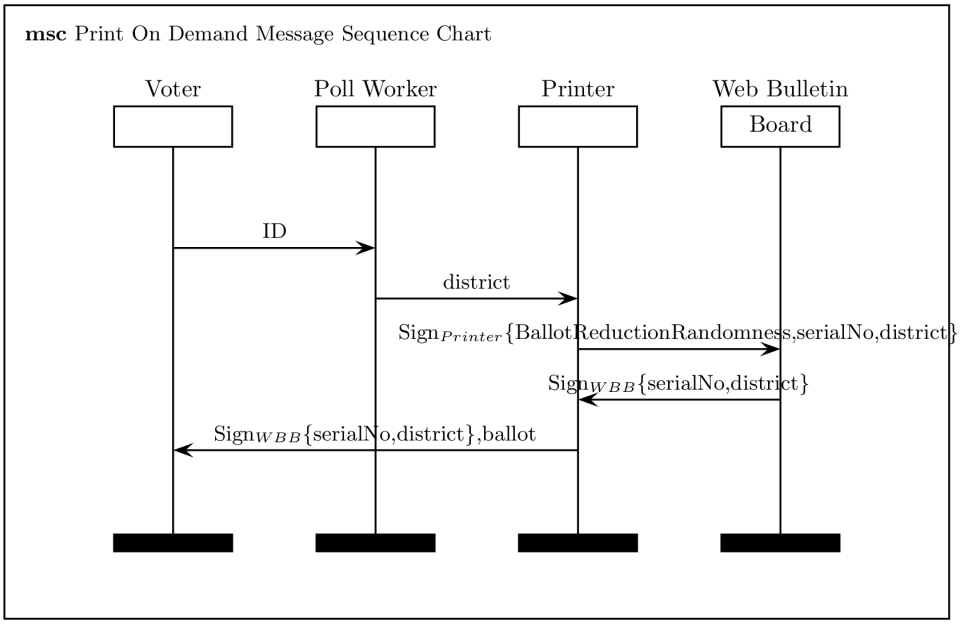

Fig. 5. Print on Demand Message Sequence Chart

but this could be detected at audit time like any other kind of bad printing. We want to be able to demonstrate afterwards on the WBB that it used the right ciphertexts. The protocol is as follows:

Pollworker:. authenticates the voters (using whatever secure or insecure method is traditional) and sends a print request to the Print On Demand device specifying the district/region they can vote in,[11]
Printer:. retrieves the next available ballot and looks up the number m of candidates in the submitted district/region.
Printer:. sends to the WBB:
— the serial number,
— the division, and
— a list *BallotReductionRandomness* of randomness values for the unused ciphertexts (i.e. the ones from $cand_{m+1}$ to $cand_n$), together with their respective permuted locations so the WBB can check them. The randomness values are those computed by the printer in the algorithm in section 2.4)
WBB:. checks that the ciphers held for $cand_{m+1}$ to $cand_n$ are encryptions of the candidate IDs of the unused candidates for the specified division.
— if valid it signs the serial number and the division and returns it to the printer, and posts the randomness values to the WBB so they can be publicly checked;
— if invalid it returns an error message.
Printer:. Checks the WBB signature of the serial number and division and, if valid, prints the ballot and signature. The printer knows the permutation and plaintexts so does not need to do any crypto to print the ballot
Voter:. votes on the ballot exactly as if it had been generated for the right number of candidates,

[11] More generally, it is the pollworkers' responsibility to authenticate the voter and request the ballot(s) that the person is eligible to vote on.

Voter:. shreds the candidate list,
EBM (or scanner if there is one):. submits the ballot to the WBB exactly as if it had been generated for the right number of candidates,
WBB:. accepts (and signs) the ballot only if it is accompanied by a signed serial number and division
EBM (or scanner):. prints the sig on the receipt,
Voter:. (optionally) checks the sig, which covers only data visible to the voter.
Voter:. later checks their vote on the WBB. They only need to check the serial number and order of their preference numbers—the correct opening of the unused (too big) candidate numbers will be universally verifiable.

3.2. (Print) auditing

Suppose a voter wants to audit a printed ballot, i.e. to check that the printed candidate list matches the ciphertexts on the WBB. The following is performed:

Voter:. requests an audit from the same printer that printed their ballot,
Printer:. sends to the WBB the randomness to open the commitments to the randomness on each CRT_i used to generate the ballot in the ballot generation phase
WBB:. checks the serial number has not already been voted on or audited and if not, opens the commitments, reconstructs the ballot, computes the permutation π, posts all the data on the public WBB, and sends a jointly signed copy of π (or candidate names in permuted order) to the printer
Printer:. The printer prints the signature
Voter:. checks the signed order of candidates π against the order printed on the ballot (note, the permutation signed by the WBB should reflect any successful ballot reduction already performed).
Voter:. takes their audited ballot home and checks that the value provided on the WBB matches the candidate order that was signed.

Figure 6 shows the message sequence chart for auditing.

3.3. Forward Secrecy

The randomness held on a printer is sufficient to reconstruct the ballots, and hence reveal the candidate orderings. If a printer is stolen, the randomness it holds will expose the associated ballot forms. Hence it is desirable for a printer to delete the randomness for ballots on which votes have been cast, since there is a potential privacy breach.

However, when a printer prints a ballot it must allow for a print audit which will require it to open the commitment to the randomness. Therefore, after a printer has printed a ballot, it must retain this randomness for some period of time.

After the ballot has been used to cast a vote, an audit is not allowed and so the randomness no longer needs to be retained and can be deleted. Deletion can be triggered by a confirmation message to the printer, signed by the WBB, when a vote is cast. Alternatively a time limit can be set on an audit request following ballot printing, and the randomness can be deleted after that time if no audit request has been received.

Note that the encrypted randomness values and symmetric key are sent directly to the printer and not posted on the WBB. As such, there is no publicly available information that could be combined with a stolen, but previously honest, printer to reveal used ballots. The symmetric keys sk_i should be deleted from the printers after the RT_i tables have been decrypted.

4. A FASTER VARIANT WITH A SHORTER PERMUTATION COMMITMENT

Although the above protocol is quite efficient, it still requires the WBB to do a lot of computation to open all relevant commitments and reconstruct the ballot permutation each time a print audit occurs.

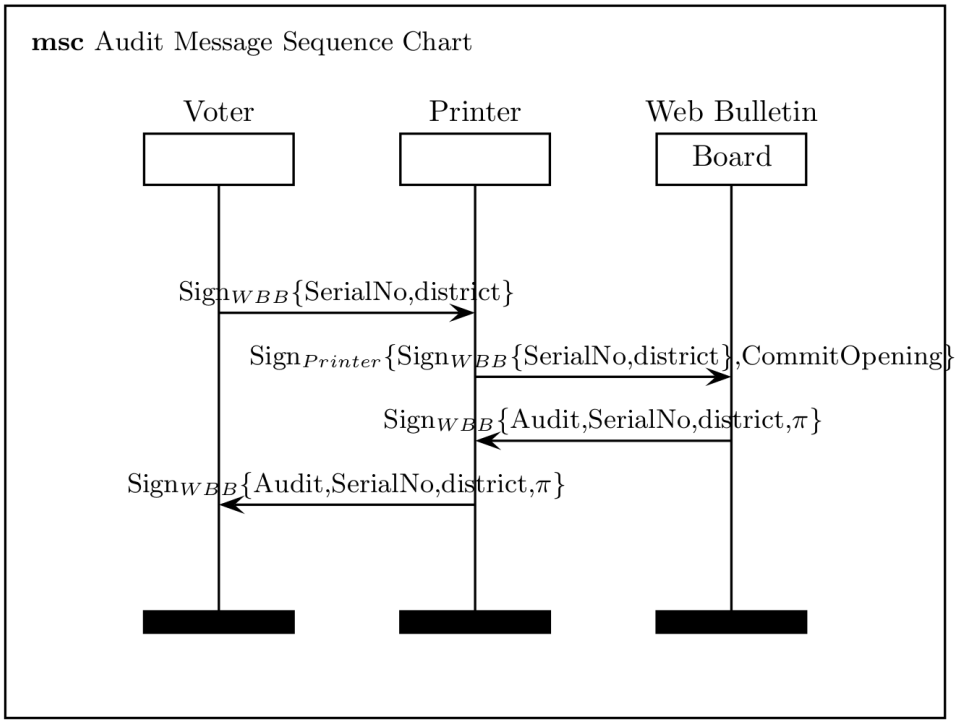

Fig. 6. Print on Demand Audit Message Sequence Chart

This is unfortunate because we would like to encourage ordinary voters to perform print audits by making them easy and fast.

One way to speed up print auditing is to ask the printer to commit to the permutation π directly when it generates the ballot, then ensure that this commitment to π is audited for proper generation (during ballot generation audits) and for conformance with the ballot permutation (during print audits). During a print audit, the WBB needs only to open and verify the commitment to π, then sign it and return it to the printer. Every print audit then triggers a ballot audit, which opens all the commitments just as described in Section 2.5, but this does not have to be done while the voter is waiting for the print audit to complete.

Like other randomness values used by the printer, the randomness used in the commitment must also be generated by the *RGen* servers, using a new column in each RT_i and CRT_i table. The printer retrieves the random value the same way that it retrieves all the others, and uses it to compute the commitment to π. That commitment is sent, along with ballot generation ciphers, to the WBB during the ballot generation stage. It is also audited, along with the ciphers, during the ballot generation audits so we gain a statistical assurance the commitments to the permutation are correct. During the print audit everything proceeds as described above, except the WBB only has to open and check the commitment and then sign the permutation based on that.

This does not affect universal verifiability, because the same data linking the ballot permutation to the commitments on the WBB is eventually published either way. However, the big advantage is it reduces the workload on the WBB during the critical time that the voter is waiting for the signature on π, since it can now defer the re-encryptions necessary to verify the permutation.

More precisely, if we add the required random values into the $n+1$-th column of each RT_i, Algorithm 1 would now be:

Algorithm 2: Deterministic Encryption by Printer with explicit WBB commitment to π.

for $i = 1 \rightarrow G$ **do** ▷ Decrypt the symmetric keys from the *RGen*Servers
  $sk_i \leftarrow Dec_{SK_P}(esk_i)$
end for
for $j = 1 \rightarrow b$ **do** ▷ b is the number of ballots.
  for $k = 1 \rightarrow n$ **do** ▷ n is the number of candidates.
    $rand \leftarrow SHA(SymmDec_{sk_1}(RT_1(j,k)) \| \ldots \| SymmDec_{sk_G}(RT_G(j,k)))$
    $CT_{j,k} \leftarrow Enc_{PK_E}(cand_k; rand)$
  end for
  $CT_{(j)} \leftarrow Sort(CT_{(j)})$ ▷ $CT_{(j)}$ returns the entire row
  $perms_j \leftarrow \pi$ ▷ π is the permutation applied to sort CT_j.
  $rand2 \leftarrow SHA(SymmDec_{sk_1}(RT_1(y,n+1)) \| \ldots \| SymmDec_{sk_G}(RT_G(y,n+1)))$
  $commit_\pi \leftarrow c(\pi; rand2)$
end for
Send CT and $commit_\pi$ to the WBB. to WBB.

Ballot generation audit (Section 2.5) would be exactly as above. Additionally, anyone can recompute the randomness value the printer used to commit to the candidate permutation π, and hence open that commitment and check that it matches the ballot permutation.

Print generation audit (Section 3.2) would be:

Voter:. returns to the same printer they previously used and requests an audit
Printer:. sends to the WBB the randomness to open the commitments to the randomness on each CRT_i used to generate the ballot in the ballot generation phase and the randomness to open the commitment to π.
WBB: (immediately). checks the serial number has not already been voted on or audited and if not, opens the commitment to π, checks it, and if valid sends a jointly signed copy of π (or candidate names in permuted order) to the printer.
WBB: (later). opens all the commitments for this ballot, reconstructs the ballot, computes the permutation π, posts all the data on the public WBB.

The final step of opening all the other commitments reduces the total number of audits that need to be done. If we relied entirely on the immediate check of the serial number and the frequency of ballot generation audits, then the system would still be universally verifiable, but the probability of the printer cheating successfully would be higher.

5. RUN-TIME FAILURES

5.1. Real-time Printer Replacement

If a printer is stolen or fails, all the ballots that had been generated by it are no longer available for use. This is important because the "printer" is a small tablet PC that would be easy to carry. Hence we need to have a suitable method for bringing replacement equipment back online during election time. For example, we could bring the randomness generation servers back online each night, or when needed, to generate new randomness, after having deleted those values they had already sent to printers. An alternative is for the randomness generation authorities to do the same thing as described above for a few extra as-yet-undeployed printers, put the data on the public WBB, and then ask each one to send their sk_i to some (distinct) entity who is going to be online at voting time.

5.2. Loss of WBB

The Print on Demand protocol requires the participation of the Web Bulletin Board. Intermittent loss of connection to the WBB will be handled by the communication infrastructure, but loss of the network or the WBB will require fallback to a mode of operation that is purely local.

Since vVote is intended to operate alongside standard paper ballots (for 2014) the fallback position will be to return to "plain EBM mode": the EBM machines will be used purely for constructing and printing a paper ballot which can then be cast with the standard paper ballots. No receipts will be issued and the vote will not be submitted through vVote. The Print on Demand service is not used at all for plain EBM voting.

6. SECURITY CLAIMS

This protocol is meant to achieve three classes of security properties:

integrity: . meaning that all attempts to manipulate a voter's input into the mix would be detected by a ballot audit, print audit, the voter's check of their printed vote, the signature check, or the final check that their receipt was properly recorded on the WBB.[12][13]
privacy: . meaning that either the printer itself or all of the randomness generating authorities must collude to reveal the contents of a vote.[14]
resistance to kleptographic attacks: . meaning that a printer attempting to leak information via the WBB data will be detected with some probability.

6.1. Integrity based on audits

An informal argument for the integrity of each person's vote is:

— The ballot-generation audit confirms that the ballot is a permutation of properly-encrypted candidate identifiers.
— The ballot-printing audit confirms that the printed list of candidate names matches the encrypted candidate identifiers on the WBB.
— The voter's check of the EBM's printout confirms that the correct numbers (or other marks) are recorded against the correct candidate names.
— The signature check confirms that the printed number sequence matches what was submitted to the WBB.
— The check of the vote on the WBB confirms that the correct ciphertexts were used (in the case of a larger-than-necessary generated ballot) and that the vote submitted to the WBB was posted.

Of course the first two audits are performed only on ballots that are *not* subsequently voted on. The argument is that any attempt to manipulate the vote by generating or printing invalid votes will be detected by random audits with some probability.

6.2. Vote privacy

Clearly if the printer leaks its information it can violate vote privacy for everyone who used a ballot it printed. This means that practical opportunities for compromising the printer must be reduced as much as possible, *e.g.* turning off the wireless connection.

Apart from the printer and an electronic ballot marker (if there is one), no other single entity can violate vote privacy.

Claim: A collusion of all but two randomness generation authorities does not have sufficient information to recover the ballot permutation (in polynomial time with non-negligible probability).

[12] Of course this does not imply that they will always be detected, if those audits are not performed on the manipulated ballot. The claim is that any manipulation can in principle be detected by an audit if it is performed.
[13] A similar claim applies to attempts to manipulate the output of the vote mixing process which happens afterwards, but that is not the topic of this paper.
[14] It's also possible for a threshold of key sharers or a sufficient number of vote mixing servers to collude to reveal a vote, but that's not part of this sub-protocol.

Clearly if all the randomness generation authorities collude and share their information, they learn the contents of all ballots. If at least two choose their random values correctly and keep them secret until the others have committed, and if the others can be forced to open their commitments, the resulting list of random values has $2k = 2 * 256$ bits of entropy.

NIST [Barker and Kelsey 2012] states that the SHA family of hashes are suitable as randomness extractors. In [Barker and Kelsey 2012] it states that when using a hash function F in which $Y = F(S||A)$ then "If the input string S was assessed at $2n$ bits of min-entropy or more (i.e., $m \geq 2n$), then Y may be considered to have n bits of full entropy output". The value of A can be anything, including null, it is just additional data. This tells us that provided at least two mix servers provide good randomness values the output from the hashes will have $k = 256$ bits of entropy.

The usual subtlety arises if we consider the possibility that some authorities might use blocking to bias the output, *i.e.* might wait until learning the other authorities' random values and refuse to open their own commitments if they didn't like the result. (This could happen, for instance, in collusion with a corrupt printer.) This is why true coin-tossing protocols are more complicated than the simple one in this proposal. In practice, such a blocking authority would be removed quickly without having the opportunity to affect many bits of the output.

Clearly the same argument holds for the PRNG construction of Section 2.4.2, given appropriate assumptions about the PRNG.

6.3. Kleptographic attacks

The output of the printers is entirely determined by the randomness that is sent to them, and other publicly committed information given in Figures 1 and 2. Hence they have no opportunity to provide any information which may be skewed in a particular way. Correct information posted therefore cannot leak information from the printer. Incorrect information will be detected with some non-negligible probability by the ballot-generation audit processes.

Although the whole group of randomness generation authorities can collude to mount a kleptographic attack, a similar argument to that for vote privacy shows that a smaller collusion has insufficient information.

6.4. Selecting Ballots for Generation Audit

Clearly it is important that we use a suitable source of randomness for the selection of the ballots to audit. Ideally a publicly verifiable source would be good. This will require further investigation to see what procedures are possible in practice.

Of course, it is difficult to compute the appropriate amount of auditing for an IRV/STV election, especially in advance [Magrino et al. 2011; Cary 2011]. This question will have to be addressed for the project, but is out of scope for this paper. We expect that most of the IRV (*i.e.* single-seat) contests will have a relatively easy margin computation in practice. However, the full STV contests are a different matter and require significant further thought. One possibility is to announce the result, explain what quantity of cheating might have been possible given the amount of auditing that was done, and ask any election challenger to demonstrate a set of votes in which they win a seat and the number of changed votes is reasonably probable given the auditing.[15]

7. CONCLUSION AND FURTHER WORK

The application of Prêt à Voter to a real election has raised interesting research questions, including those that were considered solved under different assumptions. This paper presents a print-on-demand protocol with all the important desirable features of other protocols in the literature, but with much more feasible computational cost.

It would be interesting to try to extend these ideas to schemes such as Wombat or StarVote in which the ballot is directly encrypted by the voting device. A check for the proper use of randomness could easily be incorporated into the existing voter-initiated challenges these schemes offer. Indeed,

[15]Thanks to Ron Rivest for this suggestion.

the devices would not necessarily have to be online for this, as long as they were loaded with an authenticated list of proper random values to use. It's not surprising that this can be made to work since Markpledge 2 achieves the same result, albeit with a more complex voting protocol.

8. ACKNOWLEDGEMENTS

Many thanks to Craig Burton and Zhe (Joson) Xia for helpful comments.

REFERENCES

B. Adida. 2008. Helios: Web-based Open-Audit Voting. (2008). www.usenix.org/event/sec08/tech/full_papers/adida/adida.pdf and heliosvoting.org.

Ben Adida and C. Andrew Neff. 2009. Efficient Receipt-Free Ballot Casting Resistant to Covert Channels. In *EVT/WOTE 2009, Proceedings of the Electronic Voting Technology Workshop / Workshop on Trustworthey Elections, August 10th 2009, Montreal,Canada.* http://assets.adida.net/research/adida-neff-markpledge2.pdf

Michael Backes, Dario Fiore, and Esfandiar Mohammadi. 2013. Privacy-Preserving Accountable Computation. In *Proc. 18th European Symposium on Research in Computer Security*.

Elaine Barker and John Kelsey. 2012. *NIST DRAFT Special Publication 800-90B Recommendation for the Entropy Sources Used for Random Bit Generation*. Technical Report. NIST.

Josh Benaloh, Mike Byrne, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B. Stark, and Dan S. Wallach. 2011. STAR-Vote: A Secure, Transparent, Auditable, and Reliable Voting System. (2011). http://arxiv.org/abs/1211.1904

Dan Boneh, Ben Lynn, and Hovav Shacham. 2004. Short Signatures from the Weil Pairing. *J. Cryptology* 17, 4 (2004), 297–319.

Richard Carback, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul S. Herrnson, Travis Mayberry, Stefan Popoveniuc, Ronald L. Rivest, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. 2010. Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy. In *Proc. USENIX Security*.

David Cary. 2011. Estimating the Margin of Victory for Instant-Runoff Voting. In *USENIXAccurate Electronic Voting Technology WorkshopWorkshop on Trustworthy Elections*.

Gogolewski et al. 2006. Kleptographic Attacks on e-Election Schemes. In *International Conference on Emerging trends in Information and Communication Security*. http://www.nesc.ac.uk/talks/639/Day2/workshop-slides2.pdf.

M. Jakobsson, A. Juels, and Ronald Rivest. 2002. Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. In *USENIX Security Symposium*. 339–353.

Thomas R. Magrino, Ronald L. Rivest, Emily Shen, and David Wagner. 2011. Computing the Margin of Victory in IRV Elections. In *USENIXAccurate Electronic Voting Technology WorkshopWorkshop on Trustworthy Elections*.

Silvio Micali, Michael Rabin, and Salil Vadhan. 1999. Verifiable random functions. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*. IEEE, 120–130.

Kim Ramchen and Vanessa Teague. 2010. Parallel Shuffling and its application to Prêt à Voter. In *Proc. USENIXAccurate Electronic Voting Technology Workshop*.

Alon Rosen, Amnon Ta-shma, Ben Riva, and Jonathan (Yoni) Ben-Nun. 2012. Wombat Voting System. (2012). http://www.wombat-voting.com.

P.Y.A. Ryan. 2007. *Prêt à Voter with Paillier Encryption, Revised and Extended*. Technical Report CS-TR-1014. University of Newcastle upon Tyne.

P.Y.A. Ryan and S. Schneider. 2006. Prêt à Voter with Re-encryption Mixes. In *European Symposium on Research in Computer Security (Lecture Notes in Computer Science)*. Springer-Verlag.

Daniel R. Sandler, Kyle Derr, and Dan S. Wallach. 2008. VoteBox: A tamper-evident, verifiable electronic voting system. In *Proc. 17th USENIX Security Symposium*.