Experiences Gained from the first Prêt à Voter Implementation

David Bismark, James Heather, Roger M. A. Peel, Steve Schneider, Zhe Xia
Department of Computing
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, United Kingdom
j.heather@surrey.ac.uk

Peter Y. A. Ryan
Faculté des Sciences
de la Technologie et de la Communication
University of Luxembourg
Luxembourg
peter.ryan@uni.lu

***Abstract*—Implementing an electronic voting system for the first time can be difficult, since requirements are sometimes hard to specify and keep changing, resources are scarce in an academic setting, the gap between theory and practice is wider than anticipated, adhering to a formal development lifecycle is inconvenient and delivery on time is very hard. This paper describes all of the work done by the Prêt à Voter team in the run-up to VoComp in 2007 and enumerates a number of lessons learned.**

***Keywords*-electronic voting, implementation, Prêt à Voter**

I. Introduction

The Prêt à Voter electronic voting system [1] is end-to-end verifiable, meaning that voters are able to check that their vote has been included in the announced result, and that the correctness of that result is based on all the votes cast in the election – without breaking the secrecy of the election or introducing the dangers of coercion or vote buying. The first implementation of this system was made by a group from the University of Surrey in 2006/07 in order to run a student sabbatical election and to enter the University Voting Systems Competition (VoComp) in 2007.

The key innovation of the Prêt à Voter scheme is to encode the vote using a randomised candidate list. Suppose that our voter is called Anne. At the polling station, Anne chooses at random a ballot form sealed in an envelope; an example of such a form is shown in Figure 1.

In the booth, Anne extracts her ballot form from the envelope and makes her selection in the usual way by placing a cross in the right hand column against the candidate of choice (or, in the case of a Single Transferable Vote (STV) system for example, she marks her ranking against the candidates). Once her selection has been made, she separates the left and right hand strips along a thoughtfully provided perforation and discards the left hand strip. She is left with the right hand strip which now constitutes her *privacy protected receipt*, as shown in Figure 2.

Anne now exits the booth clutching her receipt, registers with an official and casts her receipt. Her receipt is placed over an optical reader or similar device that records the random value at the bottom of the strip and records in which cell her X is marked. Her original paper receipt is digitally signed and franked and is returned to her to keep.

The randomisation of the candidate list on each ballot form ensures that the receipt does not reveal the way she voted, so ensuring the secrecy of her vote. Incidentally, it also removes any bias towards the top candidate that can occur with a fixed ordering.

The value printed on the bottom of the receipt is the key to extraction of the vote. Buried cryptographically in this value is the information needed to reconstruct the candidate order and so extract the vote encoded on the receipt. This information is encrypted with secret keys shared across a number of tellers. Thus, only a quorum of tellers acting in together are able to interpret the vote encoded on the receipt.

After the election, voters (or perhaps proxies) can visit the WBB and confirm their receipts appear correctly. Once this is over, the tellers take over and perform anonymising mixes and decryption of the receipts. All the intermediate stages in this process are posted to the WBB and are audited later. Various auditing mechanisms ensure that all the steps, the creation of the ballot forms, the mixing and decryption etc all performed correctly, but these are carefully designed so as not to impinge on ballot privacy. An accessible account of Prêt à Voter can be found in [4].

Obelix	
Idefix	
Asterix	
Panoramix	
	7304944

Figure 1. Prêt à Voter ballot form

X
7304944

Figure 2. Prêt à Voter ballot receipt (encoding a vote for "Idefix")

II. Motivation and Approach

VoComp was initially announced in September 2006. The competition required teams to submit an implementation of an electronic voting system, to prototype it in a campus-based election, and take a working demonstration to the VoComp conference [5].

The time gap between finding out about VoComp and the University of Surrey Students' Union (USSU) election that we would use in our entry was sufficiently tight that we were aware from the start that some corners would need to be cut. Indeed, the initial discussions were concerned with whether the project was feasible at all; and it was only a feeling that VoComp was too big an opportunity to miss that persuaded us that we should give it a go. In a commercial context, the project would have been deemed infeasible from the start.

Having made the decision to take the project on, several high-level design issues were resolved almost immediately:

≤ *Prêt à Voter flavour:* It was clear that we had to implement the older version of Prêt à Voter, using RSA onions [1], rather than the newer flavour using ElGamal [2]. Although this was a disappointing decision to have to make, because the ElGamal version is rather cleaner and supports extra features, we were aware that we had no off-the-shelf threshold ElGamal implementation, and it simplified things greatly to stick with the standard Sun Java SDK cryptography extensions.

≤ *Programming language:* We chose Java because we wanted the program to be cross-platform. We believed that we could demonstrate the flexibility of this by having different parts of the system running on different operating systems, in a fairly plug-and-play fashion.

≤ *Tallying method:* The USSU's constitution specifies the single transferable vote as the tallying method. It would have been far easier to implement plurality voting, but this was not an option open to us.

≤ *Easily-understood security features:* Electronic voting is difficult to trust if one's vote disappears into an obscure system, only to appear as part of a tallied result. Paper receipts from many voting machines do not secure the whole process, from the voter's point of view. The premise behind our implementation of Prêt à Voter was that if the polling booth machine was unable to determine how each voter voted, then it would not be able to delete unwelcome votes, or change them to something that it deemed more acceptable. This blindness was achieved through voters choosing their own ballot form, marking it, and only inserting one part of the form into the voting machine. The machine would have to register that vote honestly if it were to appear correctly for voter verification on the final bulletin board. This end-to-end (E2E) concept enables considerable confidence to be gained at the time of system testing - if one form can be encrypted successfully, verified by the voter in the polling booth, presented correctly on the web bulletin board, and decrypted at the time of tallying, then all of the pathways for successful votes will have been tested. If the decryption fails, something has broken and needs rectification.

There then followed what has to be described as an 'accelerated' Software Development Lifecycle (SDLC).

A. Analysis

The analysis phase was very informal and consisted mainly of a short series of meetings with University of Surrey Students' Union (USSU) officials. The main system requirements were simply those put forward by these officials, and they were usually not negotiable. We then added technical requirements of our own, which were largely related to our use of Prêt à Voter. Usability requirements ensured that our system would be understandable by the voters.

B. Design

The design process was guided by two major considerations: the short timescale, and the limited resources available (in terms of both staffing and expenditure). Many potentially fruitful avenues were closed off from an early stage because of one or both of these.

Much of the design discussion was taken up with questions of the user interface. Should we use a paper ballot form, or a point-and-click screen-based interface? Each had its drawbacks.

We opted for using paper ballots. This was certainly the correct call in terms of producing a professional system. It is possible in retrospect that it increased the workload to an intolerable level; perhaps this is a corner we could have cut but did not.

The design phase gave us a high-level overview of the system, and told us who would implement what.

C. Implementation

Although the design phase gave us seemingly clear-cut boundaries between various parts of the system, and a logical way of dividing these parts between the members of the team, it did not automatically provide the bridges between these parts. Therefore the implementations of the various parts went ahead without much consideration of how they were to communicate with each other. This later meant that we had to glue the parts together in not so beautiful ways. This was also the most important source of bugs later on. (Lesson 10)

During the implementation phase it (mistakenly) seemed to us that it would be quicker to develop the various components without certain tools needed by the election officials. To be more clear, we created a way to create and alter an election on the Web Bulletin Board system but this did not output something which could be input by the ballot form creation process or by the Optical Character Recognition (OCR) system. Therefore there was not one set of changes that needed to be made if the election changed, but several. And, as it turned out, the election specification was to change several times. (Lesson 11)

Further problems arose when the integration of the various components caused us to discover things that were lacking in the analysis and design. One such thing was the passing of error messages: if the scanning or OCR processes experienced an error of any sort, this must be output to staff and the voter. However, this meant that this error should be passed via a "vote submission" component to the "printer" component, something that had not been picked up by the previous analysis. (Lessons 12 and 13)

Finally, because we started out with little time on our hands and spent more time than we anticipated on integration issues, there was little time to test the system and to train poll workers. At this stage, we were very happy that Prêt à Voter actually verifies that the election system works without the need to verify individual components of the system.

III. Chronological sequence of events

In this section we simply tell the story of how the development of the first Prêt à Voter implementation happened and reference the next section, which enumerates the lessons we have learned.

A. VoComp announcement (spotted September 2006)

We recall the overall purpose of the competition being clear (groups of students with academic advisers build electronic voting systems and a panel of judges selects the best one) but the details less so (system requirements, judging criteria, documentation etc) and immediately after spotting the competition announcement in September or October 2006 our group started planning our entry.

The competition requirements were arguably unclear at the start but our impression is that they kept changing all the time up to the deadline. As this was the first time VoComp was held we agree it natural that a discussion must be held regarding its setup but in hindsight it was unfortunate that the competition was not more well specified at the start and we feel that the "seeking" of an optimal setup on part of the organisers and panel of judges did, in fact, continue all the way through to the final judging in Portland, Oregon.

We were tremendously impressed with the organisers of VoComp for getting the whole concept of an electronic voting competition off the ground and we do attribute our hard work in 2006/2007 to the incentive placed by the competition. (Lesson 1)

B. Agreed with USSU to run the election

In a series of meetings with the President, Deputy Returning Officer and other members of the University of Surrey Students' Union (USSU) it was agreed in October or November 2006 that the Electronic Voting Group of the Computing Department would supply an electronic voting system to be used in the sabbatical elections the following February.

No funding came from USSU but they agreed to source the perforated paper needed to print the Prêt à Voter ballot forms and to lend us a number of laptops during the election period itself.

Clear requirements.: There were only four requirements that were clearly laid forward in these meetings. (Lesson 2)

1) *STV.* USSU elections are single transferable vote (STV) elections with an extra candidate called Re-Open Nominations (popularly "RON"[1]) and no accommodation of write-in candidates. If "RON" wins a race, that part of the election has to be re-run.
2) *HRPAT.* We were told by USSU officials that they had asked the National Union of Students (NUS) whether or not they were allowed to run an electronic voting trial and that the response was that they could if a paper trail was kept as backup. After explaining how Prêt à Voter works it was agreed with USSU officials that the normal Prêt à Voter voting procedure would be followed, but instead of shredding part of the ballot form and scanning the remainder, the two parts would be stapled back together and put in a ballot box. It was also agreed that a serial number would be printed on both parts of the ballot form so that any forms that fell apart could be matched back up. We agreed to this as a plausible Human Readable Paper Audit Trail (HRPAT) which could be removed in future uses of the system.
3) *Concurrent polling stations.* The Students' Union required three concurrent polling stations (one in the Lecture Theatres complex, one in the University library and one in the Management building) and thus some mechanism was to be in place to stop students from voting in more than one place. It was finally decided that a new vote cast by a voter would replace any previous ones and we implemented such a system. As identification, we used the students' campus cards and we were given an Excel spreadsheet of all eligible voters, their names and University Registration Numbers (URN) in advance of the election.
4) *Remote voting.* A remote voting facility should exist and USSU officials would decide who could cast a vote through this interface.

Unclear requirements.: A set of implementation specific requirements that were not well defined during these meetings were:

1) *The candidates.* The number of candidates and the

[1]In some elections RON is one of the more actively campaigning candidates.

names of those candidates were not given to us as early as we would have wanted. Because this was our first version of the system, some components of the system were hard-coded (e.g. the ballot form generation and the OCR processing) which made it hard (although not impossible) to change the layout of the ballot form (not the names of candidates, but the number of candidates in a race and the number of races).

2) *Referendum questions.* Similarly, we learned that there were to be at least one referendum question on the ballot form but there ended up being two. The length of the question text for these also posed "ballot form disposition" problems for us during the development phase.
3) *Who can vote online.* We were told during the meetings with USSU that there must be a facility for off-campus students (i.e. those on their placement year) to vote remotely, that is to say over the web. Who these students were or the preferred method of contacting them with voting credentials were not specified.

C. System design

Because of the limited time available to design and implement the system – recall that the project started in October 2006 and that the USSU election was held in February 2007 – the group made a number of decisions which would cut the amount of work to be done:

1) *User interface.* Our most protracted decision was over the design of the user interface. We discussed many variants of the classic Prêt à Voter voting form that can be separated in the privacy of the polling booth, with just the voter's marks being scanned and sent to the bulletin board. Alternatives mainly revolved around computer-based on-screen interfaces.
   - ≤ A paper ballot meant having to read the form somehow, which involved either a programmable camera and a mechanism for holding the ballot paper in place, or the slowness of a scanner. Experiments with webcams in 2006 did not produce high enough image quality for reliable OCR operation, due to their low resolution, the difficulty of mounting them accurately and robustly, and the difficulties of lighting the ballot forms. Sheet-fed scanners appeared to be far more reliable, and generated nicely linear images with a good resolution and built-in monochrome conversion. Paper ballots also involved image processing and optical character recognition, areas in which our team had no experience.
   - ≤ On the other hand, it was difficult to conceive of a clean point-and-click interface that did not violate one of the key principles of Prêt à Voter: that the booth machine should not be able to work out for whom the voter cast the vote. Indirection (of the sort found in Punchscan [3]) seemed too awkward for the voter; a hybrid approach, of inserting a printed list of candidate names into a slot on the side of the monitor so that it would align with voting boxes on the screen, would have involved complex mechanical mechanisms which would have constrained our choice of computer displays and would probably have restricted the size of the forms themselves to that of one dimension of those displays. Training voters to insert the ballot forms correctly into the voting machines was also considered too difficult.

2) *Seven segment display.* With no experience of optical character recognition (OCR) in the group, we decided to use a seven segment digit layout (often found in digital alarm clocks), where four vertical and three horizontal bars are either left empty or filled out to make any digit from 0 to 9, rather than free-format text input. These 7-segment marks were used on the ballot form to allow the voter to assign a rank to each candidate. We also used a cross symbol which was either left empty or filled out in response to a referendum question. Our reasoning was that in this instance it would be easier to ask voters to use this method of writing their vote on the ballot paper than to program a system which would be more intuitive. This decision was largely validated – most voters were able to follow the example on the form and vote successfully. Many of the remainder were apparently confused by the single transferable voting system as well. (Lesson 3)
3) *RSA onions.* At the time when we were designing the implementation of Prêt à Voter the theory had moved from "decryption" onions [1] using RSA to "re-encryption" onions using ElGamal [2] or Paillier [8] encryption. However, we failed to reconcile the STV requirement with the necessity of the ElGamal onion to store the vote as a single numerical value [6]. In simple terms, the problem was that a full permutation of all candidates could not be represented as a single numerical value which could successfully be put through a re-encryption network [2]. Having decided on RSA onions, we encountered the problem of the size of the RSA payload (our onion rapidly grew too large to encrypt using RSA) and we had to encrypt the onion layer in a symmetric encryption scheme using a random key which was then embedded in the layer using the relevant RSA public key (Lesson 4). We have subsequently solved this issue in [9].
4) *LaTeX.* By generating all printed material (that is to say the ballot forms and various receipts) using the typesetting package LaTeX, we were able to generate a textual representation of what was to be printed

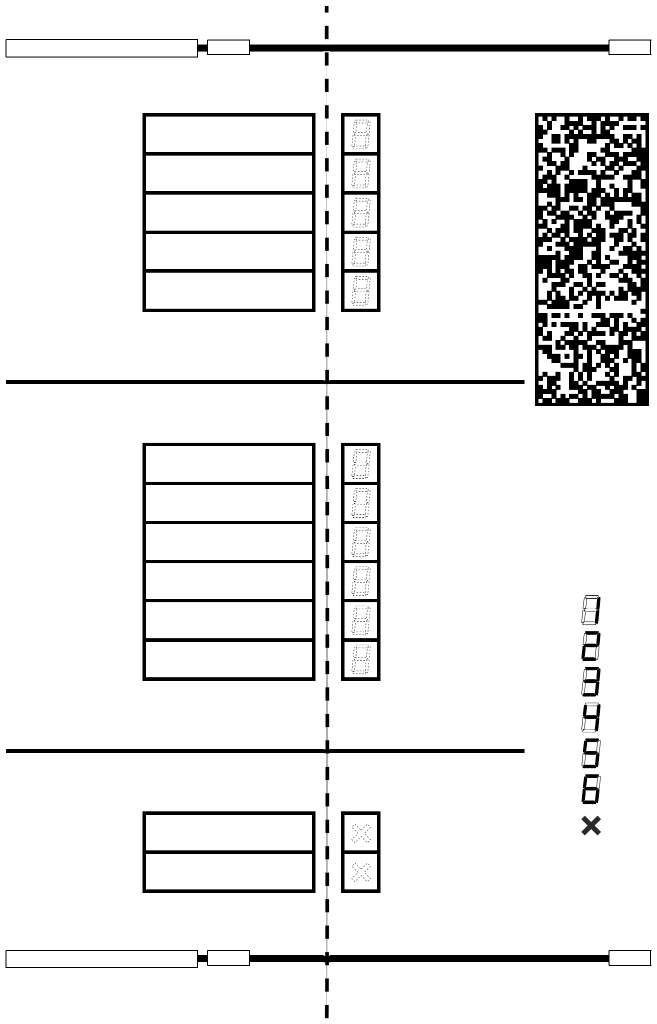

Figure 3. A VoComp ballot form; the vertical dashed line shows the position of the perforations. The left-hand side of the form is shredded, and the right-hand side (containing the voter's marks and the machine-readable form identification) is scanned.

and with a simple command combine this with the LaTeX template, which output Postscript which could immediately be sent to the printer. Developing the template was difficult, but when this was done the printing took mere seconds. However, because of the complexities of such LaTeX templates, creating a new style ballot form or one with a new layout was very difficult (and thus time consuming). An example of such a form is shown in Figure 3. (Lesson 5)

5) *Java.* Java is a versatile programming language and using it achieves not only platform independence but the possibility to compile the (open source) code using various compilers in order to achieve a higher degree of verifiability.
6) *Pre-printing ballot forms.* Running an election in Prêt à Voter version 2005 entails pre-printing ballot forms and distributing these using various procedures which ensure that the ballot secrecy is not threatened. However, because of the time constraints and the level of involvement USSU was willing to partake in, it was decided that the electronic voting group would print the ballot forms in advance and keep these safe. Thus we printed 2000 ballot forms on our Departmental high-throughput laser printer.
7) *Rudimentary poll worker interface.* We built a rudimentary poll worker interface which allowed a polling station worker not only to look up a potential voter's eligibility (searching using URN) and assign a ballot form to her, but also to initiate the scanning process with the click of a mouse button.
8) *Open source.* We believe that the code running a verifiable voting system must be made public and we thus released all code under the General Public License (GPL).

D. Coding

1) *PKI.* As Prêt à Voter uses asymmetric key encryption, not only as a basis for the encryption of the votes but also for the verification of identities of a number of distributed actors, a private key infrastructure (PKI) should be used. Such a system is used to distribute public keys together with the identities of key owners and builds up a network of trust. As there was no time to implement the use of a PKI, the group simply created the required number of key pairs.
2) *All tellers on the same machine.* The implementation has been made such that it supports any number of trusted parties (tellers) working in a distributed fashion over a computer network (possibly the Internet). In order to make the system easy to use for testing, and in order to run the system efficiently this first time, five tellers were instantiated on the same computer. Thus the web bulletin board (WBB), its database, and all of the tellers were run on the same computer even though they simulated talking to each other over a network.
3) *No diskless workstation.* Although the encrypted receipt (that remains after the candidate list is removed and shredded) is not secret, the complete ballot form is. The reason for this is that if one can see a complete ballot form before it is used to cast a vote and note down the details of it, one is able to "decrypt" the contents of that encrypted vote without knowing any encryption keys. Therefore the ballot creation should be done on what is called a "diskless workstation", that is to say a computer without a hard-drive, booted from a verified CD containing just an operating system and the ballot creation software. When the program has run and created the necessary data, the ballot forms should be sent to the printer and any technology capable of storing any secret data destroyed. All required

ballot information is stored in a database on removable media and transported to the WBB. However, because of the level at which USSU wished to hold its involvement and because of time restrictions, no such diskless workstation was set up. Instead the electronic voting group ran the ballot creation software, printed the ballot forms, destroyed the data and so forth on its own. On the other hand, we did not destroy our Departmental printer infrastructure, nor our team leader's desktop computer!

4) *SVN.* During the development phase, we used a Subversion (SVN) repository. SVN is a version control system which allowed us to share code and check in new versions of the software which was then quickly rolled out onto all development machines. We also used our SVN repository to store project and software documentation. We also stored test data for each of the interfaces between our software modules. One of the greatest benefits of the SVN repository was in the replication of software on each of the polling station computers – all of the software could be installed or updated in a matter of seconds. We did have a few last-minute problems when the software repository reported merge conflicts between different versions of the same file – we had not experienced these when we were working on our own individual components, so these issues only arose during the final stages of system integration.
5) *Not enough time for system integration.* Because of the time restriction, the different developers worked in isolation on their own parts of the system which resulted in crude connections between different parts. One example was the bridge from the OCR software, where the marks read from a ballot form were transformed into a string of comma separated values (CSV) and then transferred to a vote submission module and subsequently submitted over the network to the WBB. Whilst this worked satisfactorily, the use of XML or SOAP might have been more elegant. Protecting this information with a digital signature would also have been wise.
6) *Insufficiently modular.* Even though all of the developers were aware of the benefits of modularity in their code, there simply was not enough time to generalise modules sufficiently. Some modules that could have functioned differently are therefore instantiated, run and destroyed each time they are needed.
7) *Bug tracking.* We did not use any formal, electronic, system to keep track of bugs and open issues. Whilst bug tracking software would have been useful, it was more efficient to resolve most inter-module software integration bugs, between their respective developers, as they were discovered. This is an advantage of a small team working in a single location. (Lesson 6)
8) *Frequently changing requirements.* The requirements from VoComp and USSU often changed and we constantly chased these new requirements. Some of these changes caused major re-design work, which was difficult to manage.
9) *No code review.* We did not have time for any code reviews. Going through the code in pairs might have found simple (silly) mistakes such as the use of the Java class `Random()` instead of a secure variant.

E. Testing

1) *No structured testing.* There simply was not enough time to perform structured testing of the system. However, because of the nature of Prêt à Voter (it is fully verifiable), we were very confident that the system would collect and tally votes absolutely correctly. This is rare in systems development: we were able to formally verify the correct function of the system without verifying any code. (Lesson 7)
2) *Network testing very late.* In the USSU sabbatical election, the system would collect votes from several different locations around the Guildford campus. Whilst we had tested the network connections in the voting locations prior to the election, we encountered some contention on several network addresses when we tried to use them on the following day. This caused initial problems when opening the election (see below). We suspect that this problem was caused when another local networked facility was re-booted that morning.
3) *Never the same laptop.* Since the borrowed USSU laptops that were used in the polling stations were also used for other purposes in the Students' Union, we had to return them regularly for long or short periods. This made maintaining the software installed on each rather difficult, because we could not update them all together. Using SVN for our application code, and the Linux "yum" application for operating system updates, minimised the inconvenience.
4) *System Performance.* The polling stations comprised low-end Toshiba laptops, together with inexpensive HP 5610 printer-scanners. Each laptop could host up to eight scanners, although only one or two were used in the USSU elections. The ballot forms were scanned using the sheet feeders on the HP 5610s, and this took approximately 20 seconds per form. After this, the OCR processing took just over 10 seconds, the encryption and uploading to the web bulletin board a couple of seconds, the LaTeX receipt generation a couple more, and the printing around 10 seconds per form. The whole process therefore took around 45 seconds. For an individual form, all of these processes were sequential, but they only involved about 15 seconds of CPU time on the laptop. In principle, it

would have been possible to scan one form whilst the receipt of the previous one was being printed, but the scanner and printer units in the HP 5610 appear to be single-threaded, and so starting another scan before the printer had completed the previous receipt held up the previous voter unnecessarily. Paying more than $150 for these devices, or using separate printers and scanners, would have solved this issue.

F. Training

1) *Training USSU volunteers.* USSU volunteers were given brief training on the day before the election. They were shown how voters were to be given ballot forms, how serial numbers would be registered after URNs had been checked, how the scanning worked, and so forth.

G. Election day

1) *Remote voting was not ready.* Because of the immense pressure to complete the main system in time for the start of the election, and the lack of information on remote voting from USSU, the remote voting interface was not functional at the start of the election. There was no indication from USSU at that time that this would be a problem, in fact we believe they were preparing to send out e-mails with voting credentials after the start of the election anyway.
2) *Registration module laptop had network issues.* The first polling station that was opened by the group was the one in the lecture theatre complex. Unfortunately the laptop running the polling station administration (voter registration) module had network issues just as the election was opened. This made it impossible to record the voters as planned, but the group started keeping a paper list of voters, planning to tick these off as soon as the registration laptop was back online. A representative of the USSU voiced concerns over this but was told by the group that it did not impact voter eligibility and did not mean voters could vote more than once.
3) *Unclear start time.* Setting up the first polling station was very chaotic with lots of students, polling station workers, USSU and University officials and others in attendance. No USSU official made clear when the polling was to start and as a result everyone was confused.
4) *Mayor was delayed.* The Mayor of Guildford was invited to open the election but was delayed. This not only caused the election to be opened late, but also prevented members of the electronic voting group from proceeding to the next polling station while they were waiting.
5) *The system did work.* When eventually the election was opened, a long queue naturally built up (since lots of student voters were waiting to see the Mayor open the election and immediately queued up to vote). For about fifteen minutes, people had to stand in a queue. USSU officials may have expressed concern at this but the throughput was good, with very few votes having to be redone because they were filled out incorrectly, and soon there was no waiting time at all.
6) *Election pulled.* Despite votes now going through at a regular pace and no voter having to wait, a USSU official decided to stop the election. He said this was because other polling stations had not been opened on time. He said they would send around an e-mail to all voters saying that voting would re-start at 12.00 that same day, with USSU running the election as they had in previous years but using the Prêt à Voter random-order candidate list ballot forms so that the electronic voting group could still obtain valuable research data.

H. Scanned ballot forms

USSU allowed the electronic voting group to scan all of the ballot forms used. This meant that USSU sealed the group in a room in the Students' Union building together with USSU officials and allowed the ballot boxes to be opened. The group put adhesive tape on the back of the ballot form, over the whole length of the perforations, to allow ballot forms to be scanned without falling apart. This was also done because USSU would later cut the ballot forms up to be able to hand-tally each race separately.

1) *Hand-tallying random order ballot forms.* Because the Prêt à Voter ballot forms have a random order candidate list, they are not very easy to hand-tally in an STV election. USSU cut the ballot forms horizontally so as to divide the races and tally each race separately and concurrently to save time.
2) *Rubbish votes.* When a vote is read in by the Prêt à Voter scanner, the software checks – without knowing the contents of the vote – that the ballot form has been filled out correctly, i.e. that the vote is valid. If the voter has made an involuntary mistake, she is then able to try again. However, because these checks were not performed at the time of vote submission, there were not only a large number of spoilt votes, but also a number of the votes that USSU did allow could not be interpreted correctly by our system.
3) *Folded/stapled.* When USSU ran the polling stations, polling station workers seem to be confused as to what to do with the ballot forms (we noticed this when preparing the votes for scanning) as some were torn along the perforation, folded and stapled etc.
4) *Never processed scans.* We have not made any structured effort to process the scans, primarily because of the absence of online verification of the forms.

I. VoComp

1) *Unclear information about what was to happen at VoComp.* The information sent out to the VoComp competitors was limited, and as a result we did not know quite what to expect.
2) *Judging forms.* Just before the competition was to take place in Portland, Oregon, USA we were sent a request to create a number of judging forms. These would all be different, instructing a judge to vote in a particular way. The reason for creating these forms was that independent judges would take them and attempt to submit votes in accordance with the instructions. After the close of the competition election, the votes would be tallied and the result compared to the result intended by the judging forms. In the end, these forms were not used.
3) *Unclear judging criteria.* The judging criteria were never set out clearly in the run-up to the competition and, we believe, were not quite clear at the time of judging either. The criteria on which to base their decision were simply left up to the judges to define.
4) *Picking small holes.* One feature of the competition was the teams' chance to highlight weaknesses in the other teams' systems and to respond to critiques of their own system. As a result of there being no clear judging criteria, the teams did not know what to look for or what to highlight in the limited time available. As an example, one of our competitors found a place in our Java code where we had used the class `Random()`, which is not considered a secure pseudo-random generator. However, the fix is very quick: simply change this to `SecureRandom()`. We feel that such an easily fixed triviality was awarded greater than necessary importance.
5) *Router broke.* A short time before the start of our system demonstration at VoComp, our wireless router broke – apparently as a result of being used at 110V (despite supposedly supporting this voltage). A new router was hurriedly purchased close to the venue, and this worked without problem throughout the demonstration.
6) *Pre-printed ballot forms.* We took pre-printed ballot forms to Portland with us. We would have found it very difficult to produce modified forms at VoComp.
7) *The election worked.* Our demonstration election ran without any problem and accepted votes from all of the judges and independent testers.

J. Newcastle trial

An electronic voting trial using the Prêt à Voter implementation was made at the University of Newcastle in the AROVE-v project [7].

1) *Charity single choice election.* In order to find voters to take part in the trial, charities were invited to send people to campaign outside the polling station, drawing voters in. The charities would then be given part of a large prize sum in proportion to the number of votes they were awarded in the election. Voters could only select one charity and indicated this with an X. There were also two "referendum" style questions where voters indicated their response using an X.
2) *All asked to check their votes.* In order to test if voters were able to use the Web interface used to check one's vote, all voters were asked after voting to proceed to a row of laptops displaying the website. These were connected to the WBB laptop and so voters could immediately check the inclusion of their votes in the final tally after having them scanned.
3) *OCR failed initially.* Despite testing the system the night before, minutes before the election was due to open the OCR functionality started failing for all ballot forms. After about an hour of searching for the bug and conferring with group members on the phone it was determined that the OCR worked for previously-used test ballot forms but when new ballot forms were "freshly torn apart" there was a certain amount of fuzz at the edge of the paper. After slightly adjusting the value indicating where the OCR mechanism would expect the edge to be, scans started to be processed correctly and no more errors occurred for the rest of the election. (Lesson 8)
4) *No failed scans.* After the OCR function was properly set up and the election was opened, no scans of ballot forms failed.
5) *One scanner died.* Soon after the start of the election, one of the eight scanners failed. As the system had been built with replicated off-the-shelf hardware, the failure of one particular scanner did not impact the running of the election. In a real-world scenario, the broken scanner could immediately have been replaced with another. In this case the throughput of voters was low and therefore there was no need to replace the scanner. (Lesson 9)

After the Newcastle trial and VoComp, we have run many small demonstration elections to groups of visitors within our Department. These have validated the operational processes of initialising elections, collecting votes and generating a final, audited, tally.

IV. System Vulnerabilities

In addition to the threats and trust models of Prêt à Voter that are discussed in [4], we noticed that our implementation had some vulnerabilities that would need attention in a fuller implementation.

1) *Sensitivity to network failure* If the interconnecting computer network were to fail, our system would not function. This is because access to the web bulletin board was required for successful registration of each

vote. In addition, the ballot form auditing process required all of the teller machines to be accessible. At the end of the election, auditing and tallying also required all of the teller machines to be available and capable of handling significant workloads. We have subsequently devised a number of caching schemes to allow voting to proceed, and some thresholded cryptographic mechanisms to tolerate a small number of failed or inaccessible teller machines.

2) *Denial of Service attacks* Whilst the network could fail of its own accord, denial-of-service threats could be caused intentionally. Either the network itself, or the individual networked services, could be heavily loaded by malicious traffic, thereby slowing or halting the election process.
3) *Mind the perforations* As mentioned in Lesson 8, the paper stock used for the ballot form is critical. It has to be able to withstand scanning, which is not too difficult. It has to tear cleanly along the perforations, and this is a function of paper thickness and of the perforations themselves. If the paper is too thick, it enables the weakened perforations to provide a good separation, but laser printer toner then tends not to stick to it well, and the bending during tearing causes areas of toner to flake off the paper, thereby upsetting the scanning process. If the perforations are too strong, tearing becomes difficult. Bent, folded or torn ballots might then cause the scanner to jam. Finally, scanning ballot forms after the USSU election, after they had been stabilised with adhesive tape, emphasised that scanners can pick up dirt and debris and in sheet-feed mode this can lead to vertical stripes on the scans.
4) *Technical Support* The installation and upkeep of the polling station equipment is challenging, and requires skilled technicians (or, in our case, developers). Eventually, this skilled manpower becomes a critical resource. In a large election, especially with geographically separated polling stations, this could be a significant problem.

V. Lessons Learned

In this section we choose simply to enumerate a number of lessons that we have learned during this development. Some are common to the development of any IT system, while some are specific to electronic voting systems.

1) *Define competition rules and requirements well and early.* For optimal return on a competition like VoComp, the requirements should be drawn up, agreed and formalised before the competition commences.
2) *Sign off system requirements.* System requirements must be signed off before development starts in order to ensure that all stakeholders are not only aware of the requirements but able to see as early as possible if these meet real-world requirements.
3) *OCR.* The eventual machine interpretation of a vote written on paper by a voter should not force the voter to use a non-intuitive method for specifying the vote. An unusual marking scheme could be error-prone and cause frustration to the voters. In the case of digits or characters, it might be worth looking at open source OCR packages. However, it is also worth bearing in mind that optical character recognition is difficult – the point of CAPTCHA technology being that machines find reading text far more difficult than humans do – so the reliability of OCR decoding of unconstrained handwritten digits will never be perfect. Whether, say, 98% accuracy is sufficient is a difficult design and usability decision.
4) *Implementing theory brings surprises.* There is a considerable gap between theory and practice, meaning that any work to implement an electronic voting system which has previously only been specified theoretically will throw up issues previously not thought about. Not only might these issues take time to be discovered, but they may change the way one can actually implement the system quite dramatically.
5) *Build generalised modules.* Modules should be made as generic and as configurable as possible; i.e. developers should not hard-code any parts of the system. Whilst making a module more general is more time consuming in the development phase, this will inevitably save time whenever a change of that module is made in the future. One example of this is our ballot form printing module. In order to build a functioning system, this was initially hard-coded to work with a single ballot form, containing five races and two referendum questions. It should be clear that this style of module requires reprogramming when the number of candidates, the number of races or even the names of the candidates or races change, then any time saved in the development phase would quickly be lost. On the other hand, the production of a generic form-generation program subsequently formed the basis the whole of an M.Sc. dissertation project, and the software produced was not able to pack as many races and candidates onto a ballot form as the manual method.
6) *Use a system to track bugs and open issues.* This project was large enough that the use of a bug-tracking mechanism would have been worthwhile. We just about coped without one.
7) *Verifiable systems.* End-to-end verifiable electronic voting systems can be verified without verifying any code or procedure, because the decryption of the votes fails, or the voter checks in the polling booth or on the web bulletin board fail if the main path through all of the software processes are incorrect.
8) *Test under real circumstances.* Only if the system is

tested under real circumstances, in preparation for the opening of an election, is it possible to determine that the system is in fact working. Even then, the infrastructure might change overnight.

9) *Use off the shelf hardware.* Using off the shelf hardware in any IT system not only keep costs down, but replacement of failed components is easy.
10) *Integrate components.* Although the specification of various components of a system is important, do not forget to specify carefully (and elegantly) the way that these components fit together.
11) *Elections Change.* In the implementation of an electronic voting system for a very specific purpose, perhaps the running of a single election, it may seem possible to save time by hard-coding election definition and other variables. This is a false economy, as elections invariably change right up until the last moment that they can possibly change.
12) *Check the analysis by implementation.* The only way the analysis and design can be checked is by doing the implementation. Beware of a plethora of things popping up at this stage.
13) *Remember error handling.* In electronic voting systems, the trust in the system is crucial and therefore error handling must be very transparent. Not only does any error thrown in any component during ballot casting have to be reported and explained to the voter, they must also be logged and explained in an election log.

VI. Conclusions

This paper has reported on the first large-scale implementation of the Prêt à Voter electronic voting implementation, and the first application of Prêt à Voter to single-transferable vote elections. This development was successful due to our strong understanding of the underlying cryptographic theory, our diligent design work and a largely fault-free, if rushed, software development phase. We were able to collect votes in a large election with multiple polling stations. Subsequently, we were able to collect votes and successfully tally several other public demonstration elections.

On the basis of the final section of the paper, we conclude that there were many lessons learned: but anticipating the immense divide between the underlying theory and its practical implementation is perhaps the most important.

Acknowledgements

The authors would like to thank the organisers and judges of VoComp in 2007 for their hard work in putting the competition together and allowing us to take part in that marvellous event. It truly was memorable and we would not have missed it for the world!

We would also like to thank officials of the University of Surrey Students Union who were working under tremendous stress but were willing to put our science to a severe test.

References

[1] D. Chaum, P.Y.A. Ryan, and S. Schneider. A practical voter-verifiable election scheme. *Proceedings of the tenth European Symposium on Research in Computer Science (ESORICS'05)*, pages 118–139, 2005. LNCS 3679.

[2] P.Y.A. Ryan and S. Schneider. Prêt à voter with re-encryption mixes. *Proceedings of ESORICS*, 2006. LNCS.

[3] K. Fisher, R. Carback, and A.T. Sherman. Punchscan: Introduction and system definition of a high-integrity election system. In *PRE-PROCEEDINGS*, pages 19 – 29. IAVoSS Workshop on Trustworthy Elections, 2006.

[4] P.Y.A. Ryan. The Computer Ate my Vote. *University of Newcastle upon Tyne Technical Report no. CS-TR-988*, November, 2006.

[5] A.T. Sherman et al. VoComp Rules. available at http://www.vocomp.org/rules.php, 2007, referenced July 2009.

[6] D. Lundin. Simple and secure electronic voting with prêt à voter. *Technical Report at the University of Surrey*, CS-08-05, 2008.

[7] M. Winckler, R. Bernhaupt, P. Palanque, D. Lundin, K. Leach, P.Y.A. Ryan, E. Alberdi and L. Strigini. Assessing the usability of open verifiable e-voting systems: a trial with the system Prêt à Voter. in Kaplan, A. a. B., Asim and Aktan, Coskun and Dalbay, Ozkan (Ed.) Proc. of ICE-GOV, 12-13 March 2009, Ankara, Turkey. Vol. 1. pages 281- 296. ISBN 975-6339-00-0.

[8] P.Y.A. Ryan. Prêt à Voter with Paillier encryption *University of Newcastle upon Tyne Technical Report no. CS-TR:1014*, 2007

[9] Z. Xia, S.A. Schneider, J. Heather, P.Y.A. Ryan, D. Lundin, R.M.A. Peel and P. Howard. Prêt à Voter: All-In-One, in *Proceedings of IAVoSS Workshop on Trustworthy Elections (WOTE 2007)*, 2007, pp 47–56, Ottawa, Canada