Verifiable Classroom Voting in Practice

Feng Hao*, Dylan Clarke*, Brian Randell*, and Siamak F. Shahandashti†
*Newcastle University, UK †University of York, UK

Abstract—Classroom voting is an important pedagogical technique in which students learn by voting on the answers to questions. The same voting platform is also often used for exercises such as rating lecturer performance and voting for prizes. In this paper, we present VCV, an end-to-end (E2E) verifiable classroom voting system built based on the DRE-i protocol. Our system provides E2E verifiability without tallying authorities; it supports voting through mobile phones with constrained computing resources; it reports the tallying results instantly after voting is finished along with cryptographic proofs that enable the public to verify the tallying integrity. Since 2013, the VCV system has been used regularly in real classroom teaching, as well as academic prize competitions, in Newcastle University with positive user feedback. Our experience suggests that E2E verifiable voting through the internet and using mobile phones is feasible for daily routine activities such as classroom voting.

1 Introduction

Starting with Chaum's seminal paper published in *IEEE Security & Privacy* [4] in 2004, research on end-to-end (E2E) verifiable e-voting has become a thriving field. Informally, the notion of being E2E verifiable refers to verifiability at two levels. At the individual level, it means each voter is able to verify if their vote has been cast as intended and recorded as cast. At the universal level, it means anyone can verify if all votes are tallied as recorded. (Here we do not consider verifying the voter eligibility as that relates to voter authentication which we regard as a separate issue.) By contrast, in a traditional paper-based voting system, a voter can only check up to the point that the completed ballot is dropped into the box, not how it will be recorded and tallied in the subsequent process.

Researchers have proposed many E2E voting systems that build on a similar concept to Chaum's but improve on it in various ways. Notable E2E voting schemes include MarkPledge, Prêt à Voter, Punchscan, Scantegrity, Scantegrity II, scratch & vote, STAR-Vote, Wombat, Adder and Helios; see [7] for a review of existing systems. Some of these systems have been used in practical applications. In particular, Helios [2] was used to elect the president of Université catholique de Louvain in 2009, and since 2010, has been used for elections in universities and associations (IACR and ACM). Scantegrity [5] was adopted in the municipal elections of Takoma Park, USA in 2009 and 2011. A variant of the Prêt à Voter system [3] was used in the 2014 Victoria State election in Australia.

This paper describes a special type of E2E verifiable voting system that does not involve any tallying authorities (TAs): i.e., trustworthy individuals with computing and cryptographic expertise who are tasked to perform decryption and tallying operations. Finding such TAs in a typical classroom environment is not realistic and indeed is difficult in other voting scenarios (see [2]). We have built a concrete prototype of such a TA-free E2E verifiable system called Verifiable Classroom Voting (VCV), which uses DRE-i [10] as the underlying cryptographic protocol.

Since 2013, the VCV system has been used in real classroom teaching, as well as academic awards competitions, in Newcastle University and has received positive user feedback. The underlying DRE-i protocol is designed to be scalable to support large-scale elections [10]. It is different from boardroom voting protocols [13] in that it is centralized and does not require interactions between voters. Voters only interact with a central web server to cast votes in a way that is E2E verifiable. Although the VCV system is primarily developed for classroom voting, it can be easily applied for larger elections, such as campus voting as in Helios [2]. In the following sections, we will explain how the VCV system works, and share our experience of implementing and deploying such a system in the real-world application of classroom voting.

2 Background

2.1 Classroom Voting

Classroom voting is a pedagogical technique in which a teacher poses a question to students, allows them to think about the answer or discuss it with classmates, and then has them vote electronically on the answer. It was initially developed for use in teaching physics by Eric Mazur from Harvard University and has been developed further for use in chemistry, mathematics and other subjects [14].

Several commercial classroom voting systems are available, e.g., TurningPoint[1] (which is commonly used among UK universities including Newcastle University). However, they suffer from two main limitations. First, such systems are expensive. As well as license fees, they often require the use of proprietary hardware which may be prohibitively

This manuscript has been accepted for publication and is to appear in the IEEE Security & Privacy magazine in 2017.
Hao, Clarke, and Randell are with the School of Computing Science, Newcastle University, UK. Emails: {feng.hao, dylan.clarke, brian.randell}@ncl.ac.uk. Shahandashti is with the Department of Computer Science, University of York, UK. Email: siamak.shahandashti@york.ac.uk.

1. http://www.turningtechnologies.co.uk

expensive for many educational contexts. Second, these commercial systems do not provide verifiability; users cannot be sure that the system recorded their votes and tallied them correctly.

Our aim was to develop a practical classroom voting system that does not require any proprietary hardware and can be freely available to teachers and students, especially those in developing countries. This would overcome the first limitation, which has hampered the wider adoption of this modern pedagogy.

We also aimed to address the second limitation by leveraging the latest E2E verifiable voting techniques. This second limitation may not immediately seem like much of an issue for classroom voting. However, in practice classroom voting may also be used for such tasks as course feedback and voting for prizes for students or lecturers. Verifiability can both remove the risk of cheating from these voting sessions, and improve the trust that students have in them. Furthermore, the added assurance on the tallying integrity has the potential to greatly extend the traditional scope of classroom voting, from a pure pedagogical tool to a more general voting platform where people can exercise democratic rights on daily matters.

2.2 The DRE-i Protocol

Among existing E2E voting schemes, we choose DRE-i [10] as the underlying cryptographic protocol to build the VCV system. DRE-i is specifically designed to provide E2E verifiability *without* needing to find a set of trustworthy tallying authorities, hence the system is "self-enforcing". Furthermore, DRE-i does not require any cryptographic operations at the client side (hence obviating the need to install any Java plug-in or use JavaScript to perform expensive cryptographic operations in the browser). With the exception of digital signing, all cryptographic operations are performed before the election, thus latency during voting is minimized. Finally, in DRE-i, the tallies are *instantly* available once the last ballot is recorded, accompanied by publicly verifiable audit data to prove the tallying integrity. All these properties suit the practical requirement of classroom voting.

We provide here an overview of how the protocol works; a reader wishing further details and security proofs is referred to [10]. The central technique used by DRE-i involves pre-computing all electronic ballots in the encrypted form before the election, in such a way that the multiplication of the ciphertexts will cancel out the random factors and allow anyone to verify the tally without needing tallying authorities.

Take for example a single candidate election with "Yes" and "No" choices. The system generates $n = m \times s$ ballots before the election, where n is a product of the maximum number of eligible voters m and a safety factor s for auditing. Let $E(F_q)$ be an elliptic curve defined over a finite field F_q where q is a large prime and G be a generator for the subgroup over $E(F_q)$ of prime order p. For each ballot i, a random public key is generated: $x_i \cdot G$ where x_i is a value chosen uniformly at random from $[1, p-1]$. When this is done for all ballots, a restructured public key P_i for each ballot is computed as below:

$$P_i = \sum_{j<i} x_j \cdot G - \sum_{j>i} x_j \cdot G. \quad (1)$$

Here, P_i can be expressed in the form of $P_i = y_i \cdot G$ where $y_i = \sum_{j<i} x_j - \sum_{j>i} x_j \bmod p$. It is important to note that $\sum_i x_i y_i = 0 \bmod p$, which is called the "cancellation formula" [12]. This formula plays a critical role in the design of the DRE-i system.

For each ballot i, the value of the encrypted vote is defined as $x_i \cdot y_i \cdot G + v_i \cdot G$ where $v_i \in \{0, 1\}$ for "No" and "Yes" respectively. In addition, a non-interactive zero-knowledge proof (ZKP) is required to prove the encrypted vote is well-formed, i.e., v_i can only be either 0 or 1. This can be realized by using the standard 1-of-2 ZKP technique [6], made non-interactive by the Fiat-Shamir transformation [8]. The encrypted ballot together with the ZKP forms a cryptogram, as shown in Table 1. The first three columns are published on a public bulletin board (a publicly accessible website), while the "Yes" and "No" cryptograms in the last two columns are kept secret. Here, all public key operations are defined over an elliptic curve (we use NIST P-256) instead of a finite field as in [10]. This is to be consistent with the actual implementation in practice. The underlying DRE-i protocol is not changed.

The pre-computation generates "Yes" and "No" cryptograms that satisfy several properties. The first property is called "concealing", which means that for any selected row of Table 1, if the voter is only given a single cryptogram, she will not be able to tell if it is "Yes" or "No" since the given cryptogram is indistinguishable from random. The second property is called "revealing", which means that if both cryptograms are given, she will be able to trivially distinguish "Yes" from "No", since the encrypted value of the former is the latter adding G. The third property is called "self-tallying", which defines the relation between ballots. Suppose a single cryptogram is arbitrarily selected from every row of Table 1. Given such selected cryptograms, anyone is able to compute the tally of the "Yes" votes contained in the entire selection, though without being able to learn values of individual votes. This tallying process can be done by anyone without needing any tallying authority. For mathematical proofs of these properties, we refer the reader to [10].

Once the election setup has been done, voting is a simple two-step procedure. First, a voter chooses a candidate and receives its cryptogram printed on the receipt. Second, the voter chooses to confirm or cancel the selection. In the case of "confirm", the vote is cast and the receipt is appended with a notice that the vote has been cast. Otherwise, the vote is cancelled and the receipt is appended with the other cryptogram, the previously selected candidate and a notice that the vote has been cancelled. This allows the voter to audit whether the cryptograms have been assigned correctly to candidates. In this case, the voter is given another unused ballot to start over from step 1 (in Section 3.2, we will show an example interaction in VCV to illustrate the voting and tallying process in more detail).

Ballot No	Random public key	Restructured public key	Cryptogram of no-vote	Cryptogram of yes-vote
1	$x_1 \cdot G$	$y_1 \cdot G$	$x_1 \cdot y_1 \cdot G$, 1-of-2 ZKP	$x_1 \cdot y_1 \cdot G + G$, 1-of-2 ZKP
2	$x_2 \cdot G$	$y_2 \cdot G$	$x_2 \cdot y_2 \cdot G$, 1-of-2 ZKP	$x_2 \cdot y_2 \cdot G + G$, 1-of-2 ZKP
...	...	...	...	...
n	$x_n \cdot G$	$y_n \cdot G$	$x_n \cdot y_n \cdot G$, 1-of-2 ZKP	$x_n \cdot y_n \cdot G + G$, 1-of-2 ZKP

Table 1: Setup before the election

3 Classroom Functionality

The DRE-i protocol only provides the functionality needed to vote for one candidate out of a list. A fully featured educational polling system must handle registration, voter authentication, sessions with multiple questions, questions with different input types, the display of results, election generation and verification. We explain details of these features in this section.

3.1 Session Generation

Coordinator. In VCV, the person who creates the election is called a *coordinator*. First, the coordinator logs on to the coordinator's account on the e-voting website[2]. The authentication of the coordinator is currently through the campus LDAP service, so anyone who is a member of staff or a student of Newcastle University can log on using their campus credentials without needing to register. (We are adding external user registration facilities to make VCV publicly available under the support of an ERC Proof-of-Concept Grant).

Voting session. Often an election coordinator wants to present several questions for students to answer in one go. To allow this, we design the VCV system so that each voting session can consist of as many questions as the coordinator requires, which are presented to the voter in order.

There are four types of questions supported in VCV.

1) *Multiple choice with a single answer;*
2) *Multiple choice with multiple answers;*
3) *Free numerical input;*
4) *Free text input.*

The first two types of questions are implemented based on the DRE-i protocol. For type-1 questions, the VCV system provides one "Yes"/"No" election for each candidate and allows each voter to cast a "Yes" vote in one election and "No" votes in all of the other elections. At the verification stage it is possible to check that the number of votes cast is equal to the number of "Yes" votes in each election. The type-2 questions are handled in a similar way except that voters are allowed to cast more than one "Yes" votes. At the verification stage it is possible to check that the number of "Yes" votes is equal to or less than the number of choices allowed per voter multiplied by the number of votes cast. In this case, we no longer have a strict equality that can be used to detect stuffing of "Yes" votes, but a malicious system adding votes in this way can still be detected if the voter opts to audit the vote, as we will explain.

The last two types of questions do not use any E2E verifiable scheme. Students simply enter free inputs (either number or text), and the server displays the received inputs in the end. No audit data is provided to allow public verifiability. (We are not aware of any E2E voting system that can support these types of questions without involving tallying authorities.) Nonetheless, these types of questions are useful in practice as they allow open-ended questions to be asked to gauge student understanding and/or provoke discussion. Since the implementation of these questions is no different from existing classroom voting products, we will not discuss them further, but instead will focus on the first two types of questions.

Authentication. When creating a voting session, the coordinator can choose one of the following options for voter authentication.

1) *No passcode.* The session requires no passcode. Anyone who knows the election ID can vote.
2) *Group passcode.* The session requires one common passcode. Only those who know the election ID and the passcode can vote.
3) *Individual passcode.* The session requires a unique passcode from each individual. Each individual passcode can be used only once.

The first two authentication options are suitable for routine classroom questions where ballot stuffing is not of any concern. The group passcode is usually communicated to the class of students verbally by the teacher or through the display of a computer projector. This ensures that only the students attending the class will know the passcode. However, it does not prevent a student from sending answers multiple times using the same group passcode. The option of "no passcode" allows students to vote without any passcode. In this case, the VCV system provides a simple locking mechanism to prevent potential vandalism. The coordinator can "lock" the session right after its creation. A locked session does not accept any vote until it is unlocked by the coordinator, which is normally done during the class. This basic lock/unlock control limits the session to open only for a specific time slot controlled by the coordinator.

The third authentication option is intended to be used for voting sessions in which the "one person one vote" rule needs to be enforced (e.g., rating the lecturer's performance or voting for a prize). The system generates random individual passcodes based on the number of eligible voters specified by the coordiantor. Typically, the passcodes are printed onto individual slips of paper, folded and placed in a box, and physically mixed in front of voters in a room before being handed out to each voter. Any remaining passcodes are destroyed. This enables observers to see exactly who has been issued a passcode, without being able to link the passcodes to voters. After voting, everyone is able to check if the number of cast ballots recorded on the bulletin board matches the number of voters. In Section 4.2, we will give examples on how this has been done in practice.

2. https://evoting.ncl.ac.uk

3.2 Voting

The VCV system allows students to vote using their own mobile devices, typically smart phones, but also tablets or laptops as long as the devices are connected to the internet. In general, we provide three voting interfaces: an Android app[3], an iOS app[4] and a generic web voting page[5]. In practice, we recommend the students to use the web interface, since this avoids needing to install yet another app on their phones.

We now show an example interaction of voting through the web interface. The experience using the Android/iOS apps to cast a vote is similar. The voting procedure is basically the same as in [10], except that it is now done through the Internet in an unsupervised environment instead of using touch-screen DREs in a supervised environment at polling stations. First, the voter enters an election ID and a passcode (if any) as shown in Figure 1a. A voting session begins with a list of candidates to choose from (see Figure 1b). In this example, it is a single-answer question, so the voter is limited to select only one candidate.

Casting a vote follows two stages. In Stage 1, the voter chooses one candidate from the list. The server returns a receipt, which is a truncated hash (based on SHA-512) of the following data: the encrypted yes-vote for the chosen candidate and the encrypted no-votes for all the other candidates. The computation of the hash is *deterministic* based on the public keys published before the election. The truncated hash and the full audit data including the ZKPs and a digital signature (based on ECDSA) are published on the bulletin board. In this example, the truncated hash consists of 24 characters in Crockford's base-32 encoding[6] (see Figure 1c).

In Stage 2, the voter is prompted to confirm or cancel this selection. To minimize the interactions, this step is integrated with the display of the stage 1 receipt as shown in Figure 1c. If the voter opts to cancel, then the server issues the second part of the receipt: a notice that this is a cancelled vote along with the name of the cancelled candidate; see Figure 1d. Meanwhile, this receipt and the full audit data including the encrypted no-vote for the chosen candidate, the encrypted yes-votes for all the other candidates, the ZKPs and a digital signature, are published on the bulletin board. The voter can choose to start over with a new ballot or skip answering this question. The voter can cancel up to s ballots, where s is a safety factor that is configurable during the election setup (by default, $s = 5$). On the other hand, if the voter chooses to confirm, the server records the vote and issues the second part of the receipt: a notice that the ballot is confirmed (see Figure 1e, 1f, 1g for an example of steps to cast a confirmed vote). This receipt, together with a digital signature, is published on the bulletin board. In all figures, the "Verify Your Receipt" button is a hyperlink that points to the public bulletin board where the same content of the given receipt is published.

In VCV, a *public bulletin board* is simply implemented as publicly readable web pages on which the server publishes audit data with accompanying digital signatures to prove the data authenticity. Any voter who wishes to audit the system performs auditing as follows. At Stage 1, the voter checks if the given receipt matches what is published on the bulletin board. At Stage 2, when cancelling the vote, the voter checks if the same receipt is published on the bulletin board, along with the cancelled candidate name that should match the voter's selection. In the case of confirming the vote, the voter checks if the given receipt matches what is published on the bulletin board. Out of concern for usability, these verification options are made *available* to voters, but not imposed on them as required steps.

3.3 Tallying and Verification

Once all students have cast votes, the coordinator closes the voting session using the web portal. The tallies are instantly available (Figure 1h) since the server directly records votes according to DRE-i. Meanwhile, any unused ballots are published on the bulletin board with both "Yes" and "No" cryptograms revealed and a note to indicate that they are unused. These, together with the previous confirmed and cancelled ballots, and full digital signatures, are available on the bulletin board and can be downloaded as a single XML file. The XML format allows a computer program to process the file and verify all data in a batch operation. On the bulletin board, an open-source verification program written in Java is also provided, but anyone can write their own verification programs.

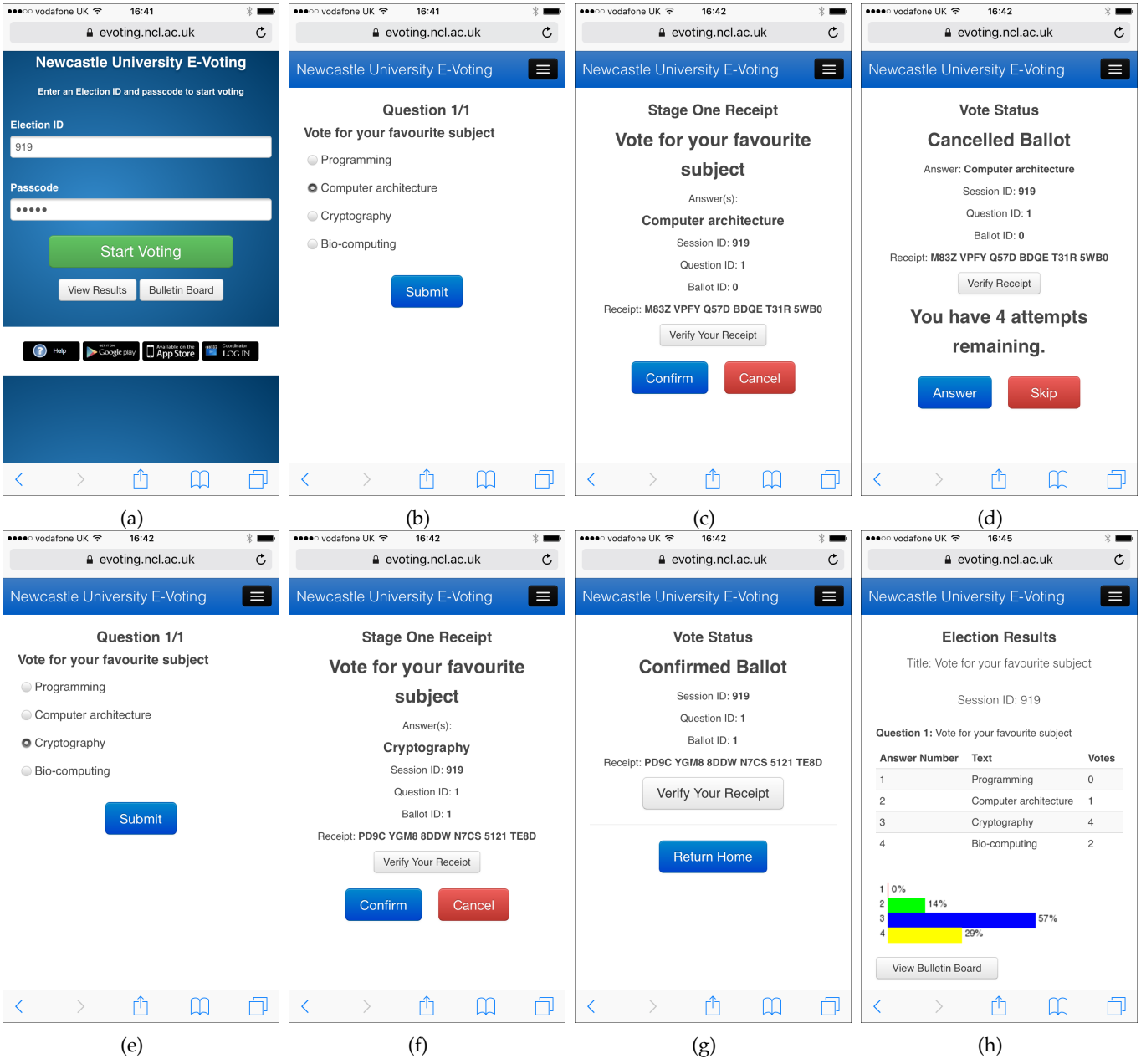
3.4 Other Functions

We now briefly explain some other functions that we have found useful in practice. First, we find that users sometimes wish to use the same session (or very similar sessions) on multiple occasions – for example, a lecturer may want to reuse the same voting questions as in previous years. Hence the session generation application allows for an existing session to be duplicated, and for this duplicated session to be changed before it is created. Second, for each created voting session, the VCV system generates a downloadable PDF file with all questions and answers in a format that can be displayed in a slide show through a computer projector. This serves as a useful fall-back mechanism in the case that the Internet is down and VCV cannot be used. With the saved PDF slides, the lecturer will still be able to practise the classroom voting pedagogy, albeit in an old-fashioned way by asking students to raise hands to vote for answers.

3.5 Security and Privacy Considerations

There are several security and privacy considerations that need to be highlighted. First, by design, the DRE-i web server records the vote directly (similar to a Direct Recording Electronic machine). Hence, the privacy of the voter relies on physical or procedural means to ensure the voting is anonymous: i.e., the server does not know the voter's real identity. Second, although DRE-i is free from tallying authorities, it requires all audit data to be available on the bulletin board in order to verify the tally (because the cancellation of random factors involves all ballots). This adds

3. https://play.google.com/store/apps/details?id=uk.ac.ncl.evoting
4. https://itunes.apple.com/us/app/newcastle-university-evoting/id565080670
5. https://evoting.ncl.ac.uk
6. http://www.crockford.com/wrmg/base32.html



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 1: An example interaction of voting through the web interface

to the robustness requirement for a practical implementation. Finally, the VCV system is implemented for internet voting in an *unsupervised* environment. It does not provide resistance against coercion, e.g., a coercer may stand over the voter's shoulder. For this reason, VCV is only suitable for low-coercion elections (as is the case for Helios [1]).

4 Practical Experience

In this section, we share our experience of implementing and deploying the VCV system for practical use.

4.1 A Pedagogical Tool for Classroom Teaching

Building on an early implementation of DRE-i for internet voting and trials in 2011 [11], we developed the first prototype of the VCV system with built-in functions for classroom voting in 2013. The system was initially available to only a few lecturers in the School of Computing Science, at Newcastle University, and the student feedback on the pedagogical value of VCV was overwhelmingly positive. For example, in a survey conducted in 2013 among a class of MSc students, 95% of the 20 respondents considered that voting made the lecture more fun, and 100% of them agreed that voting helped them learn. Subsequent surveys indicate similarly positive feedback [9].

Encouraged by this positive student feedback, we continued to use the VCV system for the following academic years in 2014, 2015 and 2016, and plan to continue its use in future. In 2014, the system was extended to be made available to the campus of Newcastle University; any member who has a campus account can log on to the e-voting server and create elections. With the positive student feedback

that confirms the pedagogical value of classroom voting, the VCV system has been adopted by lecturers in other schools (e.g., EEE and Business) at Newcastle University in their classroom teaching. Currently, we are extending the VCV system to make it publicly available to users outside Newcastle University.

4.2 Academic Awards Competition

Besides its use in classroom teaching as a pedagogical tool, the VCV system has been regularly used for academic competitions with a prize for the winner. For this kind of application, the option of "individual passcodes" should be used and the proper physical procedure followed in distributing passcodes in order to ensure "one person one vote". As an example, on 10 February 2016, over 50 members of the Secure & Resilient Systems (SRS) group in the School of Computing Science, Newcastle University, used the VCV system in the annual "Best Paper Award" competition to vote for their favourite paper written by a PhD student or Research Associate within the group during 2015. Individual passcodes were printed on paper slips, folded and physically randomized before being handed out to people who were attending the competition event. Voting started after a series of presentations by candidates who submitted their papers for this competition. The results were instantly announced in the presence of all attendees in the room with the winner being awarded an Amazon voucher of £200. Similar voting competitions based on VCV were held in other research groups by following a similar procedure, e.g., the formal methods and the computational biology groups.

The VCV system has also been used in some university events. On 18 November 2015, in a Research Impact in SAgE (RISe) event organized by the Faculty of Science, Agriculture & Engineering at Newcastle University with over 150 attendees including many external to the university, the VCV system was used to let attendees vote for their favourite presentation for the "Impact in Progress" award. The voting results were instantly announced on the spot with the winner being awarded a cheque of £1,000. The use of democratic voting for selecting the winner, the convenience of using VCV for that purpose and the instant availability of results with public audit data to prove the tallying integrity were commented on positively by the attendees.

4.3 Truncated Hashed Receipts

The use of a truncated hash for the receipt, instead of the full hash or full cryptographic data, is motivated by an observation in [11] that most users found it difficult to compare two long strings of random-looking cryptographic data (in base-64 encoding). Therefore, to make it easier for a voter to verify the receipt, we chose to use a short truncated hash with the first n characters given to the voter. The truncated hash is encoded using Crockford's base-32 instead of base-64 to make the receipt not only human readable, but also pronounceable (which will allow integrating VCV with accessibility tools such as a screen reader for visually impaired voters in our future developments.)

Clearly, using the truncated hash and base-32 encoding presents a trade-off between security and usability. We begin by noting that the public keys are generated at random before the election and are made public in advance. The restructured public keys can be subsequently computed by anyone. A base-32 encoding with length n gives us 32^n possible values. With $n = 24$ (default value used in the system), that corresponds to $32^n = 32^{24} = 2^{120}$. Each full receipt has the ballot number of each cryptogram and the surrounding XML fully specified. Hence, the only way that the system can generate another valid receipt is to swap one or more cryptograms in the receipt.

We first consider the case with type 1 questions. Here, exactly one answer must be chosen, so any valid receipt must contain one "yes" cryptogram and "no" cryptograms for every other answer. Hence, the only changes that can be made to a valid receipt (without turning it into an invalid receipt) is to swap the yes cryptogram for the corresponding no cryptogram and swap one of the no cryptograms for the corresponding yes cryptogram. If there are m answers then there are $m - 1$ ways this can be done.

Under the random oracle model we assume that the hash value returned for each of these $m - 1$ receipts is random. Hence, given these $m - 1$ receipts, the probability of one of them giving the same receipt value as the original receipt is $\frac{m-1}{32^n}$. For $n = 24$ and $m = 5$, this gives the probability of 3.0×10^{-36}.

We now consider the case with type 2 questions. Here, we do not have the case where exactly one answer must be chosen, with instead any number of answers between 1 and some upper bound being allowed. Hence, if there are m answers then, in the extreme case where the upper bound is m, the system can produce $t = \sum_{i=1}^{m} \binom{m}{i} = 2^m - 1$ valid receipts, giving a probability of $\frac{t-1}{32^n}$ that a different receipt has the same truncated hash as the original receipt. With $n = 24$ and $m = 5$, the probability is 2.2×10^{-35}.

In the analysis above, we have assumed that the election setup is done honestly. However, it is possible that a dishonest server may repeatedly try the setup until it finds collisions in the truncated hash. With $n = 24$, it will require 2^{120} operations, which is computationally infeasible. If the attacker has a vast amount of memory, he might try the birthday attack, which will require 2^{60} operations to find a collision (with 50% chance of success). This will involve supplying inputs of the encrypted votes to the hash function. In the simplest (2-candidate) case, the input contains a pair of encrypted votes with the minimum size of 66 bytes (compressed points over the NIST P-256 curve). This attack will require a memory space of $2^{60} \times 66$ bytes: that is 76 million terabytes. This is beyond the capability of an ordinary attacker. We note that the VCV system is designed only for low-sensitive elections.

In conclusion, we find that $n = 24$ is a suitable value for providing minimum required security for common classroom voting scenarios without imposing too much strain on voters to verify the receipt. In the election setup, the value of n is configurable by the coordinator. For highly sensitive elections, we recommend that a longer length should be chosen. On the other hand, for those types of elections, a polling station based implementation in a supervised environment [15] would be more appropriate than our internet-based VCV system.

Table 2: Ballot Generation Time as Number of Answers Varies

Questions	Answers Per Question	Voters	Time (sec)
1	2	50	39.6
1	3	50	68.0
1	4	50	81.8
1	5	50	108.6

Table 3: Ballot Generation Time as Number of Questions Varies

Questions	Answers Per Question	Voters	Time (sec)
1	4	50	81.8
2	4	50	163.3
3	4	50	246.3
4	4	50	328.0

4.4 Performance

We analyse the performance of two functions of the system: election generation, and verification of election results. Vote casting is not analysed as this involves no cryptographic operations other than the production of digital signatures, and does not result in any noticeable delay to users.

Ballot Generation. The ballot generation time is measured on a virtual machine (VM) with access to one 2.8 GHz core and 4 GB memory running CentOS, where the VCV web server is currently hosted.

The ballot generation time is shown as number of questions varies in Figure 2, as number of answers per question varies in Figure 3 (a minimum of two answers is required for a multiple choice question), and as number of voters varies in Figure 4. In each case, the question generated was of type 1, although the ballot generation for type 2 questions should perform the same operations. Taken together, these figures show that ballot generation times are approximately linear in the product of the number of questions, the number of voters and the total number of answers (that is, the number of actual ballots that need to be generated). They also show that elections for reasonable class and session sizes can be generated quickly using standard hardware.

Verification. The verification of election results can be done by anyone. In our experiment, the time of verification was measured on a 2.8 GHz dual-core PC with 4 GB memory running Windows. Verification times were calculated as one parameter was varied: the number of answers. As verification involves checking figures for each ballot in turn, verification time should depend upon the number of ballots, whether they are varied through increasing the number of answers, the number of questions or the number of voters. We fixed the number of questions at 1, the number of voters at 50 and the number of available ballots (to allow for auditing) at 5 per answer per voter.

The verification procedure involves the following steps. 1) (*Tally*) The cryptograms published on the bulletin board are multiplied (according to DRE-i) and the result is checked to be equal to $t_i \cdot G$, where G is the base point on the elliptic curve and t_i is the tally for answer i; 2) (*ZKP*) all zero-knowledge proofs are checked to be valid; 3) (*Digital signature*) all digital signatures are checked to be valid; 4) (*Audit*) for cancelled and unused ballots, the relationships between the two cryptograms are checked to be correct, and additionally, for cancelled ballots, the stage-1 receipt matches the voter choice that is revealed in the stage-2 receipt; 5) (*Keys*) the restructured keys are checked to be correctly generated from the public keys for each answer. The times taken for verification as the number of answers varies are shown in table 5.

Table 4: Ballot Generation Time as Number of Voters Varies

Questions	Answers Per Question	Voters	Time (sec)
1	4	50	81.8
1	4	75	121.6
1	4	100	165.7
1	4	125	204.2

Table 5: Verification Time (sec) as Number of Answers Varies

Answers	Tally	ZKP	Signature	Audit	Keys
2	0.05	117.5	8.9	0.06	0.11
3	0.09	171.8	9.0	0.07	0.14
4	0.13	228.7	9.1	0.08	0.18
5	0.16	287.1	9.2	0.09	0.22

These values show us three key facts. First, the time taken to verify the proofs is much more significant to the overall verification time than the other three values combined. Second, the verification time increases in an approximately linear manner as the number of answers, and hence ballots, increases. Third, a normal sized session can be verified in a reasonable space of time on a home computer.

Low-spec server. The VM that hosts the web server has limited resources, however it is still sufficient for the campus use. Largely, this is because the VCV system pre-computes all cryptographic encryptions before the election, so the latency during voting is minimized. Typically, a lecturer creates a voting session one day before the class to allow plenty time for the pre-computation to finish. The latency during voting is the most sensitive to users; in practice, it is almost negligible even when many voters are using the system simultaneously.

4.5 Lessons Learned and Future Work

STV. The first lesson we learned is the need for supporting versatile election schemes. In 2014, the Newcastle University Students Union (NUSU) contacted us and expressed a strong interest to use the VCV system for their presidential elections. The union elections had been using paper ballots based on Single Transferable Vote (STV), but many voters were dissatisfied with the manual tallying process which was error-prone and slow. However, VCV was found unsuitable for their purpose, since it did not support STV and the union did not want to change the tradition of using STV. How to extend VCV to support STV is a research challenge that we are working on.

Usability. A second lesson concerns the usability. Some users asked us to remove the confirm/cancel step to make the system more usable. In one example, during the voting competition at the RISe event in 2015, we observed that, when explaining to the audience in the room how to vote, the host explicitly asked people to choose "confirm" in step 2. The option of confirm/cancel in step 2 is to enable voter-initiated auditing, allowing the voter to check if the system

is behaving honestly. In practice, we find that ordinary voters seem more willing to trust the system or expect others to audit the system rather than performing auditing by themselves. This shows a subtle gap between what researchers want voters to do and what voters actually want to do. Bridging this gap is non-trivial and needs further research.

Other areas. We also highlight a few other areas that we plan to improve in future work. First, the option of "individual passcodes" currently requires a physical procedure to distribute the passcodes, which is not always convenient. We plan to add support of an electronic distribution mechanism, e.g., following a similar email-based method to that used in Helios [1]. Second, the current VCV system does not provide dispute resolution. If a voter claims the receipt does not match, it is difficult for a third party to judge if the server is misbehaving or the voter is being dishonest. This seems an inherent issue for internet voting and demands further research to address it. Third, although the pre-computation approach has the advantage of minimizing the voting latency, it also limits the voting schemes to simple ones like voting from a pre-determined candidate list. In future work, we plan to incorporate more E2E voting protocols into the VCV platform to cover a wider range of voting schemes.

5 Conclusion

We have presented, to our knowledge, the first practical verifiable classroom voting (VCV) system. We have detailed the functionality needed to turn a verifiable e-voting protocol into a fully functioned verifiable classroom voting system, which has been used in real classroom teaching with positive student feedback. We intend to continue with further work to improve functionality and usability, as well as making the system publicly available.

Acknowledgement

This work is supported by ERC Starting Grant (No. 306994) on "Self-Enforcing E-Voting" and ERC Proof-of-Concept Grant (No. 677124).

References

[1] Ben Adida. Helios: Web-based open-audit voting. In *USENIX Security Symposium*, volume 17, pages 335–348, 2008.

[2] Ben Adida, Olivier De Marneffe, Olivier Pereira, Jean-Jacques Quisquater, et al. Electing a university president using open-audit voting: Analysis of real-world use of helios. *EVT/WOTE*, 2009.

[3] Craig Burton, Chris Culnane, and Steve Schneider. vvote: Verifiable electronic voting in practice. *IEEE Security & Privacy*, 14(4):64–73, 2016.

[4] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE security & privacy*, 2(1):38–47, 2004.

[5] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *Proceedings of the Conference on Electronic Voting Technology*, EVT'08, pages 1–13, Berkeley, CA, USA, 2008. USENIX Association.

[6] Ronald Cramer, Ivan Damgard, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In YvoG. Desmedt, editor, *Advances in Cryptology - CRYPTO 94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer Berlin Heidelberg, 1994.

[7] Feng Hao and Peter Y A Ryan (Eds). *Real-world Electronic Voting: Design, Analysis and Deployment*. Series in Security, Privacy and Trust. CRC Press, 2016.

[8] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology - CRYPTO '86*, pages 186–194, London, UK, UK, 1987. Springer-Verlag.

[9] Feng Hao, Dylan Clarke, and Carlton Shepherd. Verifiable classroom voting: Where cryptography meets pedagogy. In Bruce Christianson, James A. Malcolm, Frank Stajano, Jonathan Anderson, and Joseph Bonneau, editors, *Security Protocols Workshop*, volume 8263 of *Lecture Notes in Computer Science*, pages 245–254. Springer, 2013.

[10] Feng Hao, Matthew N Kreeger, Brian Randell, Dylan Clarke, Siamak F Shahandashti, and Peter Hyun-Jeen Lee. Every vote counts: Ensuring integrity in large-scale electronic voting. *The USENIX Journal of Election Technology and Systems*, pages 1–25, 2014.

[11] Feng Hao, Brian Randell, and Dylan Clarke. Self-enforcing electronic voting. In *Security Protocols XX: 20th International Workshop, Cambridge, UK, April 12-13, 2012, Revised Selected Papers*, volume 7622, pages 23–31. Springer, 2012.

[12] Feng Hao and Piotr Zieliński. A 2-round anonymous veto protocol. In *Security Protocols Workshop 2006*, pages 202–211. Springer, 2009.

[13] Dalia Khader, Ben Smyth, Peter YA Ryan, and Feng Hao. A fair and robust voting system by broadcast. In *5th International Conference on Electronic Voting (EVOTE)*, volume 205, pages 285–299, 2012.

[14] E. Mazur. *Peer Instruction: A User's Manual*. Series in Educational Innovation. Prentice Hall, 1997.

[15] Siamak F Shahandashti and Feng Hao. DRE-ip: a verifiable e-voting scheme without tallying authorities. In *European Symposium on Research in Computer Security*, pages 223–240. Springer, 2016.