

Chapter 13

DRE-i and Self-Enforcing E-Voting

Feng Hao

School of Computing Science

Newcastle University, UK

feng.hao@ncl.ac.uk

CONTENTS

13.1	Introduction	344
13.2	Dining Cryptographers Problem	345
13.2.1	Description of the Problem	345
13.2.2	Chaum's Original Solution: DC-net	345
13.2.3	Limitations of DC-net	346
13.2.4	First Attempt on a New Solution	347
13.2.5	Improved Solution: AV-net	349
13.2.6	Presentation at SPW 2006	351
13.3	Boardroom Electronic Voting	352
13.3.1	Open Vote Protocol	352
13.3.2	Extension to Multi-Candidate Election	353
13.3.3	Presentation at WISSec 2009	354
13.4	Large-Scale Electronic Voting	356
13.4.1	From Decentralized to Centralized	356
13.4.2	Trade-off	356
13.4.3	Direct Recording Electronic with Integrity	357
13.4.3.1	Setup	357

	13.4.3.2	Voting	359
	13.4.3.3	Tallying	360
	13.4.4	Publication of the DRE-i Paper	361
13.5		Trial Elections	362
	13.5.1	Prototyping DRE-i	362
	13.5.2	Favorite Chocolate Election	363
	13.5.3	Favorite Cheese Election	363
	13.5.4	ERC Starting Grant on Self-Enforcing E-Voting	367
	13.5.5	A Verifiable Classroom Voting System	368
	13.5.6	Cryptography Meeting Pedagogy	371
13.6		Conclusion	373
		Acknowledgment	374

13.1 Introduction

In this chapter, I will describe a journey of exploring a verifiable e-voting system that can be deployed in practice. This journey started in 2005 when I was a second-year PhD student working under the supervision of Ross Anderson in the security group, Computer Laboratory, University of Cambridge. The liberal research environment in the Computer Lab encouraged PhD students to freely explore topics of their interest, not necessarily confined by their original PhD proposal. While working on “biometric encryption” (which was my original PhD topic), I became interested in cryptography and wanted to learn more in this field.

One particular cryptographic problem that caught my interest was the “Dining Cryptographers problem,” which was first introduced by David Chaum in 1988 [139]. In 2005, Piotr Zieliński and I proposed an efficient solution, called Anonymous Veto network (AV-net). The AV-net protocol proves to be more efficient than other solutions in all three aspects: the number of rounds, the computational load and the message size. As it turns out, seemingly different problems in cryptography are often inherently related. We soon discovered that the fundamental technique involved in solving the Dining Cryptographers problem could be applied to tackle other cryptographic problems, one of which was electronic voting. With my collaborators, we extended the Anonymous Veto network protocol to small-scale boardroom voting [291] and then further adapted the boardroom voting protocol to a centralized setting to make it suitable for large-scale elections [288]. The result is an end-to-end (E2E) verifiable voting protocol called direct recording electronic with integrity (DRE-i). In contrast to other E2E voting systems that require tallying authorities (TAs), DRE-i does not involve any TAs.

The design of the DRE-i protocol shows that the involvement of TAs is not indispensable to achieve E2E verifiability. This leads to the creation of a new category of voting systems that are E2E verifiable and also TA-free. We name this new category “self-enforcing e-voting” (SEEV). In 2013, we received a €1.5m European Research Council (ERC) Starting Grant (in which I am the principal investigator) to further the

investigation on SEEV. More details about the research results and the experience of trialling SEEV systems in real-world applications will be explained in this chapter.

Contrary to the custom of thanking people in the end of an article, I would like to express my sincere thanks to my collaborators here. This journey has been greatly helped by a few people, especially: Piotr Zieliński, Peter Ryan, Matthew Kreeger, Brian Randell, Dylan Clarke, Siamak Shahandashti and Peter Lee. Without their valuable inputs (which I will explain in more detail), the results would not have been attainable.

In the following sections, I will start the journey by first describing a well-known problem in cryptography, called the “Dining Cryptographers problem.”

13.2 Dining Cryptographers Problem

David Chaum is well-known for making several seminal contributions to cryptography. In the e-voting field, two of his papers are especially influential. The first is a visual cryptographic voting scheme based on a touchscreen direct recording electronic (DRE) machine [143]. This work inspired a genre of subsequent voting schemes that are built on a similar concept but improve Chaum’s original scheme in various aspects, e.g., Prêt à Voter, Punchscan, Scantegrity and Scantegrity II. The second paper is about a mixing technique, known as mix-net [153], which involves a chain of mixing servers to randomly permute the inputs such that the relationship between the inputs and outputs is unknown unless all servers are corrupted. This mix-net concept has been widely adopted in many e-voting systems to protect voter anonymity.

As compared to extensive studies on Chaum’s above two papers, another paper of Chaum’s has received much less attention from the voting community. That paper first introduces the “Dining Cryptographers problem” [139].

13.2.1 Description of the Problem

As described in the original 1988 paper [139], three cryptographers sit around a table in a restaurant for dinner. A waiter informs them that the dinner arrangements have already been paid for, but the identity of the payer is unknown. This leads to one of two possibilities: either National Security Agency (NSA) has paid for them or one of the cryptographers has paid without telling others. The three cryptographers respect each other’s right to make an anonymous payment, but they want to find out if NSA has paid.

13.2.2 Chaum's Original Solution: DC-net

Essentially, the Dining Cryptographers problem requires a secure multi-party computation (MPC) protocol on a Boolean-OR function: given a binary input of '1' or '0' (which correspond to "I paid" or "I did not pay," respectively) from each participant, the protocol allows participants to compute the Boolean-OR of all input bits without revealing the value of each individual bit. If every participant sends '0', the Boolean-OR will be '0' which means none of the participants paid (so NSA must have paid). On the other hand, if one or more participants send '1', the Boolean-OR of inputs will be '1' which means NSA did not pay.

In the same paper [139], Chaum proposed a solution, called the Dining Cryptographers Network (or DC-net). The DC-net protocol works in two stages. In the first stage, each two cryptographers establish a 1-bit secret, say by tossing a coin behind a menu. In the second stage, every cryptographer publicly announces the exclusive-OR (XOR) of the two secret bits that he holds if he did not pay, or the opposite of the XOR if he did pay. After the second stage, every cryptographer computes the XOR of the three announced bits. If the result is '0', it means no one has paid (so NSA must have paid); otherwise, it indicates one of the cryptographers has paid, but the identity remains unknown to the other two. This protocol can be generalized to n participants where $n \geq 3$. All n participants form a fully connected graph with each person being a vertex and each shared secret key an edge. Figure 13.1 illustrates how DC-net works with an example of five participants.

The Dining Cryptographers problem is essentially the same as the "anonymous veto problem" in the MPC literature [113], except that in the latter an input bit of '1' is interpreted as "veto" and '0' as "not veto". Several anonymous veto schemes [270, 345, 113] proposed in past research can be applied to solve the Dining Cryptographers problem. However, those techniques are generally complex, lacking the simplicity and elegance of Chaum's original DC-net solution.

13.2.3 Limitations of DC-net

Although DC-net has been commonly regarded as a classic technique in cryptography, it has not been used in practical applications. Further analysis shows that this technique has several major drawbacks. First, the pairwise keys are complex to set up. Given n participants, the total number of pairwise keys has the complexity of $O(n^2)$. Second, message collision is problematic. If two participants, or in the general case any even number of participants, send the '1' message ("I paid") at the same time, their messages will cancel out each other (see Figure 13.1(d)). Chaum calls this a "collision" and suggests to resolve this problem by retransmission [139]. However, the exact retransmission mechanism is not specified. Third, the message can be easily jammed. The participant who chooses to announce his bit last can trivially suppress any messages sent by previous people. In the paper [139], Chaum acknowledges this

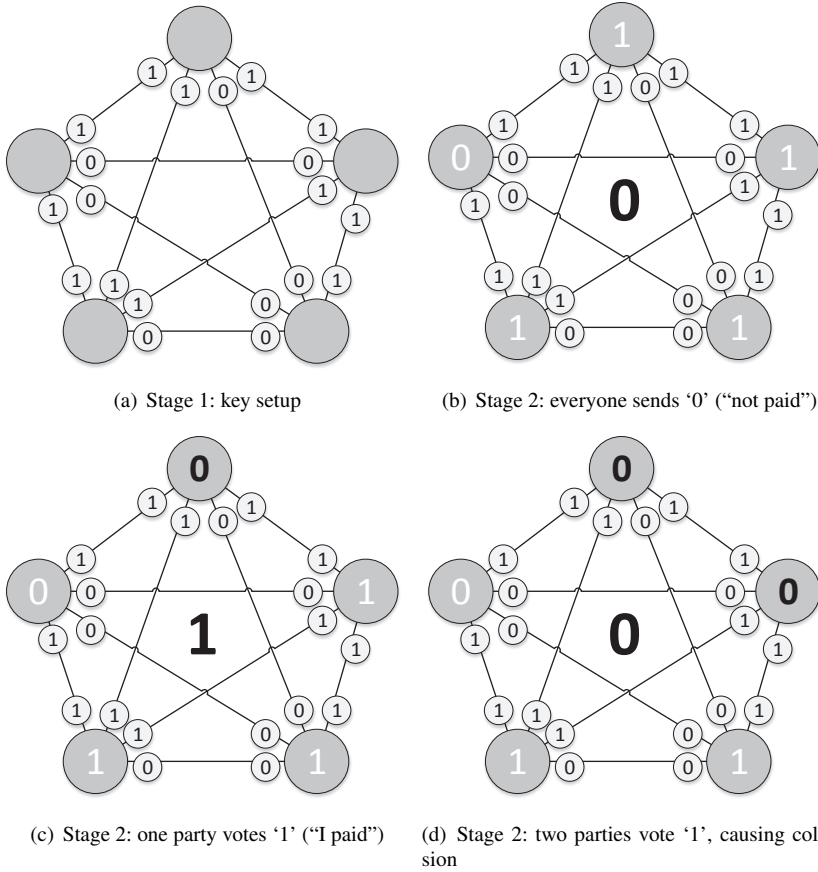


Figure 13.1: An illustration of the DC-net protocol for five participants. A collision occurs when an even number of participants send the '1' messages at the same time.

attack as a "disruption." He suggests catching the disrupter by setting up traps, but this can make the system rather complex.

13.2.4 First Attempt on a New Solution

I was motivated to find a solution to address the limitations of DC-net. After numerous failed attempts, I happened to find that by juggling random public keys in some particular order, it seemed to be able to perform secure multi-party computation on the Boolean-OR function. The result is a two-round cryptographic protocol that performs secure MPC on the Boolean-OR function. This initial protocol has never been published. As I will explain, what was published later [292] is a more efficient variant

of this protocol. Nonetheless, I think the original solution is still interesting enough that I will take the opportunity to describe it here.

First, we need to define a finite cyclic group suitable for cryptography. For example, we can use the same group setting as DSA [546]. Let p and q be large primes such that $q | p - 1$, and g be a generator of the subgroup of Z_p^* of prime order q . Assume there are n participants in the network ($n \geq 3$). The task is to enable these participants to securely compute a Boolean-OR function while preserving the privacy of each individual input.

The protocol works in two rounds. Here, one “round” refers to a step in the protocol, in which operations from all participants can be performed simultaneously without depending on each other. In the protocol description below, all modular operations are performed with respect to modulus p , so unless stated otherwise, the explicit “mod p ” notation is omitted for simplicity.

Round 1 Every participant P_i selects two random values $x_i, y_i \in_R \mathbb{Z}_q$ and broadcasts g^{x_i} and g^{y_i} .

Round 2 Every participant P_i broadcasts $A_i = (\prod_j g^{x_j})^{a_i}$ and $B_i = (\prod_j g^{y_j})^{b_i}$ where $a_i = y_i$ and $b_i = x_i$ if the participant sends ‘0’ or uses random values for a_i and b_i if the participant sends ‘1’.

$$(a_i, b_i) = \begin{cases} (y_i, x_i) & \text{if } P_i \text{ sends '0',} \\ (r_i, s_i), \text{ where } r_i, s_i \in_R \mathbb{Z}_q & \text{if } P_i \text{ sends '1'.} \end{cases}$$

After the second round, each participant compares $\prod_i A_i$ and $\prod_i B_i$ (with respect to modulo p). If they are equal, it means all participants sent ‘0’; otherwise, it means one or more participants sent the message ‘1’. However, the identities of those participants who sent ‘1’ remain unknown.

The correctness of the protocol is easy to verify. If all participants have sent ‘0’, we have

$$\prod_i A_i = \left(\prod_i g^{x_i} \right)^{\sum_i y_i} = g^{\sum_i x_i \cdot \sum_i y_i} \pmod{p},$$

$$\prod_i B_i = \left(\prod_i g^{y_i} \right)^{\sum_i x_i} = g^{\sum_i y_i \cdot \sum_i x_i} \pmod{p}.$$

Obviously, $\prod_i A_i = \prod_i B_i$ if everyone sends ‘0’. On the other hand, if one or more participants send ‘1’, the newly added randomness will break the equality.

The essence of this protocol is very similar to the original DC-net design. Both work through the cancellation of random factors. In DC-net, if every cryptographer sends ‘0’ (i.e., “I did not pay”), all random secrets used for encrypting the inputs bits will cancel each other out. This makes the final XOR result be ‘0’ (i.e., no cryptographer has paid). Similarly, in the above two-round protocol, if every participant

sends ‘0’, the random secret factors, namely x_i and y_i , will cancel each other out. As a result, $\prod_i A_i / \prod_i B_i = g^0 = 1$.

One key difference between the two solutions is that DC-net does not use public key cryptography, while the new scheme is built on public key cryptography. The use of public key cryptography allows the application of well-established zero knowledge-proof (ZKP) primitives to enforce all participants to honestly follow the protocol specification, hence effectively addressing the “disruption” attack [139]. So far for the simplicity of description, I have omitted the ZKPs in the above protocol specification, but I will highlight their important role in the next section after presenting the final version of the protocol.

One distinctive advantage of the new scheme is that it requires only two rounds. This is more efficient than all other anonymous veto protocols [345, 270, 113] that use public key cryptography. In fact, it can be proved that the 2-round efficiency is the best achievable for the secure MPC on the Boolean-OR function [294].

13.2.5 Improved Solution: AV-net

I was keen to share this finding with colleagues in the security group and get their feedback. A few people became interested. In particular, Piotr Zieliński, another PhD student in the same group, was attracted to this problem. Zieliński was a mathematician by training. To my surprise, he quickly came up with a clever improvement.

Zieliński’s improvement was to optimize the random-factor cancellation process. In my original solution, the cancellation works based on the following equation (when everyone sends ‘0’):

$$\left(\prod_i g^{x_i}\right)^{\sum_i y_i} / \left(\prod_i g^{y_i}\right)^{\sum_i x_i} = 1 \quad (13.1)$$

Zieliński quickly pointed out that it was sufficient to use just one variable to achieve the same cancellation effect. The key element in his proposal is the following proposition:

Proposition 13.1 Cancellation Formula

Given $x_i \in_R \mathbb{Z}_q$, $y_i = \sum_{j < i} x_j - \sum_{j > i} x_j \pmod q$ for $i = 1, \dots, n$, $\sum_i x_i y_i = 0 \pmod q$.

The above proposition is called the “cancellation formula” [292, 291, 288]. As will become clear later, this simple-looking formula proves to be incredibly powerful. While a mathematical proof about this proposition can be found in [292], Table 13.1 gives an intuitive illustration on the correctness of this cancellation formula.

The protocol was subsequently revised based on using the new cancellation formula. This constitutes the final version of the protocol published at Security Protocols

Table 13.1: An illustration of the cancellation formula for five participants [292].

	x ₁	x ₂	x ₃	x ₄	x ₅
x ₁		-	-	-	-
x ₂	+		-	-	-
x ₃	+	+		-	-
x ₄	+	+	+		-
x ₅	+	+	+	+	

Note: The summation $\sum_{i=1}^n x_i y_i = \sum_{i=1}^n x_i (\sum_{j=1}^{i-1} x_j - \sum_{j=i+1}^n x_j)$ is the addition of all the cells, where +, - represent the sign. They cancel each other out.

Workshop (SPW’06) [292]. A complete description of the final protocol, including the zero knowledge proofs, is given below (also see [292]).

Round 1 Every participant P_i selects a random secret $x_i \in_R \mathbb{Z}_q$ and publishes g^{x_i} and a zero-knowledge proof for proving the knowledge of the exponent x_i .

When this round finishes, each participant P_i computes

$$g^{y_i} = \prod_{j=1}^{i-1} g^{x_j} / \prod_{j=i+1}^n g^{x_j}$$

Round 2 Every participant P_i publishes $A_i = (g^{y_i})^{c_i}$ and a zero-knowledge proof for proving the knowledge of c_i , where $c_i = x_i$ if P_i sends ‘0’ or a random value otherwise.

$$c_i = \begin{cases} x_i & \text{if } P_i \text{ sends ‘0’}, \\ r_i \in_R \mathbb{Z}_q & \text{if } P_i \text{ sends ‘1’}. \end{cases}$$

After the second round, each participant computes $\prod_i A_i$. If everyone sends ‘0’, we have $\prod_i g^{c_i y_i} = \prod_i g^{x_i y_i} = 1$. On the other hand, if one or more participants send the message ‘1’, we have $\prod_i g^{c_i y_i} \neq 1$. Under the Decision Diffie–Hellman assumption, the two messages, $g^{x_i y_i}$ and $g^{r_i y_i}$, are indistinguishable [291]. Thus, the Boolean-OR function is computed securely without revealing each individual input.

In the protocol, senders must demonstrate their knowledge of the discrete logarithms without revealing them: more specifically, the knowledge of x_i in Round 1 and the knowledge of c_i in Round 2. This can be realized by using the Schnorr non-interactive zero-knowledge proof [517, 147], a standard primitive in cryptography. The use of the ZKPs ensures that all participants honestly follow the specification, hence greatly restricting the freedom of an active attacker.

Zieliński’s improvement is significant in two aspects. First, by using the new cancellation formula, every participant needs to generate just one ephemeral public

key, instead of two. The computational load is reduced by half, as is the size of the transmitted data. Second, after the improvement, the protocol becomes “ultimately simple” — i.e., as simple as possible, but not simpler. The “simplicity” of a protocol is a powerful feature. It not only facilitates security analysis of the protocol, but also has a direct impact on efficiency. As detailed in [292], the improved protocol proves significantly more efficient than related techniques [113, 270, 345] in all three aspects: the number of rounds, the computation load and message sizes. In fact, this exceptional efficiency was not the design goal, but a natural outcome of our striving for “simplicity.”

Apart from the efficiency aspects, the most critical consideration of a security protocol is whether it is secure. We tried our best to learn from the past why many protocols failed, so to make sure the same mistakes would not be repeated in our case. The understanding of previous attacks is helpful and necessary, but not sufficient to ensure a protocol secure against (undiscovered) attacks. The necessity of being able to mathematically “prove” the security of a protocol naturally arose as compelling.

In the end, we proved that the protocol was secure based on the assumption that certain number theoretical problems (particularly, discrete logarithm) were intractable. Essentially, the security of the protocol was reduced to solving the underlying number theoretical problems. In other words, breaking the protocol would imply immediate solutions to those problems, which elite mathematicians have been trying hard to solve for hundreds of years but to no avail. This reductionist proof greatly solidified our confidence in the security of the protocol.

When the design work was completed, we gave the protocol a name called “Anonymous Veto network” (AV-net) and tried to get it published.

13.2.6 Presentation at SPW 2006

We first submitted the paper to a major cryptographic conference. All reviewers seemed to like the protocol, but in the end they decided not to accept the paper as they found “no practical value” in solving the Dining Cryptographers problem.

We were disappointed, but found it difficult to disagree. In fact, I could not see any practical value either. Nonetheless, it was an interesting problem, which we enjoyed solving. That was perhaps what really mattered. Still, subconsciously I felt there should exist “practical value” somewhere, but I needed to investigate further.

We then submitted the paper to the Security Protocols Workshop (SPW’06), held locally at Cambridge. We were pleased that our paper was accepted. While presenting the paper in the workshop, I wanted to make the Boolean-OR computation problem sound more practically relevant, so I came up with the following puzzle.

A Crypto Puzzle

During an open meeting, the Galactic Security Council must decide whether to invade an enemy planet. One delegate wishes to veto the measure, but worries that such a move might jeopardize the relations with some other member states. How can he veto the proposal without revealing his identity?

The puzzle and our solution seemed to have attracted significant interest from the audience. Various aspects of the AV-net protocol were queried by experienced cryptographers present at the workshop, but after active probing, no weakness of the protocol was identified (see the transcript of discussion at [293]).

Near the end of the Q&A, Bruce Christianson, a professor from the University of Hertfordshire and one of the organizers of the workshop, raised an interesting question (see transcript [293]):

Bruce Christianson: *“Suppose we’re voting on whether to admit someone to our club, and it requires two no votes to blackball, can we generalise this approach in that way?”*

The essence of the question concerns extending the AV-net protocol to perform a Boolean-tallying function. More formally, given a binary input of ‘1’ or ‘0’, the protocol should allow n participants to securely tally the number of ‘1’s while preserving the privacy of each individual input. This is essentially a boardroom voting problem [344, 270]. One may interpret the input bit ‘1’ or ‘0’ as “Yes”/“No” in a single-candidate election.

13.3 Boardroom Electronic Voting

Christianson’s question made us realize that the research project was unfinished. We were joined by Peter Ryan who became interested in this problem too. Three of us started to work together, trying to extend AV-net to a more general boardroom voting protocol.

13.3.1 Open Vote Protocol

While the question was clear, the solution had remained elusive for quite some time. As it turned out, the key to the solution is a 1-out-of-2 ZKP technique, due to Cramer, Damgård and Schoenmakers (also known as the CDS technique) [176]. With the understanding of the CDS technique, the answer to Christianson’s question gradually became clear.

The result was a new protocol called “Open Vote.” For simplicity, we first consider a single-candidate election. Each voter casts either “0” or “1” (which corresponds to “No” or “Yes”). The tally is represented by the count of the “Yes” votes. The protocol operates in the same group setting as AV-net. It runs in the same two rounds.

Round 1 Every participant P_i selects a random value $x_i \in_R \mathbb{Z}_q$ and publishes g^{x_i} together with a zero knowledge proof for proving the knowledge of x_i .

When this round finishes, each participant P_i computes

$$g^{y_i} = \prod_{j=1}^{i-1} g^{x_j} / \prod_{j=i+1}^n g^{x_j}$$

Round 2 Every participant P_i publishes $A_i = g^{x_i y_i} g^{v_i}$ together with a zero knowledge proof showing that v_i is one of the two values $\{1, 0\}$.

$$v_i = \begin{cases} 1 & \text{if } P_i \text{ votes "yes"} \\ 0 & \text{if } P_i \text{ votes "no"} \end{cases} \quad (13.2)$$

After the second round, everyone is able to tally votes. To tally the “yes” votes, each participant, or in fact anyone observing the protocol, can compute $\prod_i A_i = \prod_i g^{x_i y_i} g^{v_i} = g^{\sum_i v_i}$. The value $\sum_i v_i$ on the exponent is the tally of “yes” votes. The equality holds because of the same cancellation formula as is used in AV-net, namely $\sum_i x_i y_i = 0$ (see Proposition 13.1). Since $\sum_i v_i$ is normally a small number, it is not difficult to compute its value by using exhaustive search or Shanks’ baby-step giant-step algorithm [361].

As in AV-net, senders must produce valid zero knowledge proofs to prove that they follow the protocol specification honestly. In the first round, each participant needs to demonstrate the knowledge of the exponent, which can be realized by using the same Schnorr’s technique [517]. In the second round, each participant needs to demonstrate that the encrypted vote is one of the two values $\{1, 0\}$ without revealing which one. This can be realized by using the standard CDS technique [176].

13.3.2 Extension to Multi-Candidate Election

So far we have only considered a single-candidate election. Obviously, if there are only two candidates, then the same single-candidate protocol is still applicable — instead of selecting “Yes”/“No”, the voter choose “Candidate A”/“Candidate B”.

To support more than two candidates, there are a few methods proposed in the literature [76, 177]. A straightforward way is to run the single-candidate protocol in

parallel for k candidates. Each voter casts a “Yes”/“No” vote to each of the candidates. The tallying for each candidate is done in parallel.

A second method, based on [177], assumes k independent generators, g_1, g_2, \dots, g_k , one for each candidate respectively. The first round remains the same. In the second round, each participant sends $g^{x_i y_i} \cdot \rho_i$ with a 1-out-of- k zero knowledge proof showing that ρ_i is one of $\{g_1, g_2, \dots, g_k\}$ (using the generalized CDS technique [176]). For tallying, one computes $\prod_i g^{x_i y_i} \cdot \rho_i = g_1^{c_1} \cdot g_2^{c_2} \dots g_k^{c_k}$ where c_1 to c_k are the counts of votes for the k candidates correspondingly.

A third method is to adopt an encoding scheme defined in [177]. Assume there are n voters. Compute m so that m is the smallest integer to satisfy $2^m > n$. Now the encoded value is defined 2^0 for the first candidate, 2^m for the second candidate, 2^{2m} for the third candidate, and so on, up to $2^{(k-1) \cdot m}$ for the k th candidate. In other words, redefine Equation 13.2 as:

$$v_i = \begin{cases} 2^0 & \text{if } P_i \text{ votes candidate 1} \\ 2^m & \text{if } P_i \text{ votes candidate 2} \\ \dots & \dots \\ 2^{(k-1)m} & \text{if } P_i \text{ votes candidate } k \end{cases}$$

The tabulation is basically the same as before: $\prod_i g^{x_i y_i} g^{v_i} = g^{\sum_i v_i}$. The super-increasing nature of the encoding ensures that the total $\sum_i v_i$ can unambiguously be resolved into the tallies for k candidates, respectively. In other words, we have $\sum_i v_i = 2^0 \cdot c_1 + 2^m \cdot c_2 + \dots + 2^{(k-1)m} \cdot c_k$, where c_1 to c_k are the counts of votes for the k candidates correspondingly.

Among the three methods, the first one is the simplest, while the other two are more complex in terms of exhaustive search. Given n votes, k candidates and that each vote is cast to one of the k candidates, the maximum number of tries for the exhaustive search in the first method is $k \times n$. In comparison, for the other two methods, the maximum number of searches for determining the tallies is $\binom{n+k-1}{k-1} = O(n^{k-1})$ (see the Combinations with Repetitions problem [538]). This is less scalable than the previous $k \times n$, but exhaustive search may still be feasible when k is relatively small.

13.3.3 Presentation at WISSec 2009

When designing the boardroom voting protocol, we followed the same design principle as before: i.e., striving for “simplicity.” Again, this leads to exceptional efficiency. The Open Vote protocol proves to be more efficient than other boardroom voting protocols [270, 344] in all three aspects: the number of rounds, computational load and message sizes [291]. The key in achieving such efficiency is attributed to the use of the “cancellation formula” (Proposition 13.1).

Besides efficiency, the protocol also enjoys two attractive theoretical features. The first is “self-tallying.” There are no tallying authorities involved at all. The sec-

ond is “maximum voter privacy.” The privacy of the voter is protected at the maximum level – only a full-collusion attack that involves corrupting all other voters can compromise the voter’s privacy. (We refer the reader to [291] for security proofs.) With the security proofs completed, we felt we had finally addressed Christianson’s question with an affirmative answer.

The paper was accepted by the journal *IET Information Security* for publication. In the meantime, I was invited to present the paper at the 2009 Benelux Workshop on Information and System Security (WISSec’09). The workshop produced no proceedings, but provided a good opportunity for researchers to exchange ideas and share their findings. This workshop was held at the Université catholique de Louvain (UCL), Belgium, where the famous Helios election was conducted a year earlier to elect the university president. This made the workshop particularly relevant to e-voting research.

While presenting the paper at the workshop, I modified the crypto puzzle at SPW’06 to fit the new problem of secure computation of a Boolean-tallying function.

A Crypto Puzzle - Follow-up

In the Galactic Republic, the chancellor is seeking re-election in the Senate. Some delegates do not want to vote for him, but worry about the revenge. All communication is monitored. There is no secret talk between delegates. In addition, no trusted third parties exist. How to arrange a voting such that the voters’ privacy will be preserved?

This new crypto puzzle generated quite some interest from the audience. However, the real inspiration of the workshop turned out to be a following presentation on Helios. The paper is titled “Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios” due to Ben Adida, Olivier de Marneffe, Olivier Pereira and Jean-Jacques Quisquater [46]. Helios is a verifiable remote voting system, which was adopted in 2008 in Université catholique de Louvain to elect the university president. It was a real-world application of e-voting and was commonly considered a milestone in the field. I was impressed by the Helios achievement and its impact on the field. While sitting in the audience to hear about the Helios work, I could not help asking myself a question.

Me: “*Can we extend the Open Vote protocol to do a similar campus election as Helios?*”

This question made the journey move on to the next goal: a practical e-voting system that can be used in real-life applications.

13.4 Large-Scale Electronic Voting

The Open Vote protocol is designed only for *small-scale* boardroom elections; extending it to support *large-scale* elections presents a non-trivial challenge. Matthew Kreeger, a colleague at Thales E-Security, became interested in this problem. Two of us then started working together to explore a solution.

13.4.1 *From Decentralized to Centralized*

The Open Vote protocol is a decentralized voting system. There is no central election authority. The election is essentially run by the voters themselves. This naturally requires cooperative interactions between voters. In the best case, two rounds of interactions are needed, as the case of the Open Vote protocol.

However, a decentralized design has critical weaknesses, which make it unsuitable for any large-scale election, such as the one conducted by Helios [46]. First, the multi-round interactions between voters make the process vulnerable to disruptions (i.e., denial of service attacks). If some voters drop out between rounds, the election will be seriously disrupted and may have to be restarted. Second, in the decentralized setting, every voter independently manages a secret key and performs cryptographic operations. This assumes that voters have a high level of expertise (or have a commonly trusted application to do it for them). Third, exhaustive search is required to determine the tally. While this is feasible in a single-candidate election, the computation can become very expensive in a multi-candidate election especially when there are many candidates.

To extend the Open Vote protocol to support “large-scale” elections, it is essential to change its decentralized structure to a centralized one for better scalability and robustness. Of course, a change in a security system is never free, as it involves trade-offs.

13.4.2 *Trade-off*

The main trade-off concerns voter privacy. Instead of voting in a decentralized way, voters now cast votes through a centralized interface, e.g., a touchscreen DRE (Direct Recording Electronic) machine at a polling station. This implies that, “maximum voter privacy,” a theoretically attractive property of the Open Vote protocol, will be lost. When a voter selects a choice on the touchscreen, the machine inevitably learns the voter’s choice. (However, this does not necessarily mean the voter’s privacy must be compromised, since her identity remains unknown to the machine if an anonymous voting procedure is followed.)

In the centralized setting, a critically important requirement of a voting system is to ensure the “integrity” of the election result. The standard method of assuring “integrity” is to allow voters to verify the result at two levels. At an individual level,

every voter should be able to verify that her vote has been recorded correctly (through voter-initiated auditing) and has been included into the tallying process (through checking the receipt against a bulletin board). At a universal level, every voter should be able to verify the integrity of the tallying result (by cryptographically checking the data published on the bulletin board). Systems that satisfy both levels of verifiability are commonly known as being end-to-end (E2E) verifiable.

We observed that E2E voting systems proposed in the past generally involved a set of trustworthy tallying authorities (TAs) to administrate the tallying process. TAs are assumed to be selected from different parties with conflicting interests and are subject to a cryptographic threshold-control scheme, e.g., based on Shamir's secret sharing [546]. In addition, TAs are assumed to be able to "independently" manage their private keys and write their own "trusted" software to perform tallying tasks. However, as highlighted in the campus election using Helios, the practical implementation of TAs can be "a particularly difficult issue" [46].

We wondered if the involvement of TAs was really necessary. To prove it was not, we set out to design a voting protocol that is E2E verifiable and TA-free. The result was a new protocol called direct recording electronic with integrity (DRE-i). This protocol was designed based on adapting the previous Open Vote protocol from a decentralized setting to a centralized one, while preserving an important property: namely "self-tallying." I will describe the details of the DRE-i protocol in the next section (also see [288]).

13.4.3 Direct Recording Electronic with Integrity

The DRE-i protocol consists of three stages: setup, voting and tallying. The protocol itself is applicable to either local voting at a polling station or remote voting over the internet. Here I will focus on describing it in the context of the former using a touchscreen DRE machine as the voting interface and then explain how it can be adapted to remote voting.

13.4.3.1 Setup

The setup is done before the election. It involves 1) setting up the system's private signing key, 2) preparing the electronic ballots and 3) publishing commitment data on a public bulletin board. For simplicity, it is assumed that the system consists of one DRE machine, but it can be easily generalized to include more machines.

First, the DRE machine generates a private signing key, say using ECDSA [546]. The public key is published on the bulletin board (a publicly accessible website).

Subsequently, the DRE machine pre-computes n electronic ballots where n is the product of the maximum number of eligible voters and a safety factor (> 1). This safety factor is to allow the generation of extra ballots for auditing purposes, as I will explain later. Typically, a safety factor is a value between 5 and 10.

For each of the n ballots, the machine generates a random private key $x_i \in_R [1, q - 1]$ and computes its corresponding public key g^{x_i} . When this has been done for all n ballots, the machine computes:

$$g^{y_i} = \prod_{j < i} g^{x_j} / \prod_{j > i} g^{x_j} \quad (i = 1, \dots, n)$$

Here, the result g^{y_i} is called a restructured public key. It is computed by multiplying all the random public keys before i and dividing all the random public keys after i (as in the Open Vote protocol [291], but all keys are generated in a centralized process).

For a single candidate election, each ballot contains two values: “Yes” and “No”. The ciphertext for the yes-vote is $g^{x_i y_i} \cdot g$, and for the no-vote is $g^{x_i y_i}$. In addition to the ciphertext, a zero-knowledge proof (ZKP) is needed to prove that the ciphertext is well-formed – i.e., the ciphertext is indeed in the form of $g^{x_i y_i} \cdot g^{y_i}$ where y_i is one of the two values $\{0, 1\}$. We can use the 1-out-of-2 ZKP [177] for this purpose.

Here, we define “cryptogram” as the combination of an encrypted vote and its associated ZKP. Hence, a yes-cryptogram refers to the combination of the yes-vote $g^{x_i y_i} \cdot g$ and the associated 1-out-of-2 ZKP. Similarly, a non-cryptogram refers to the combination of the no-vote $g^{x_i y_i}$ and the corresponding 1-out-of-2 ZKP.

Overall, the pre-computation generates a table (Table 13.2) that satisfies the following four properties.

1. *Well-formedness.* Given any cryptogram in the table, anyone is able to verify that it is an encryption of one of the two values: “Yes” and “No” (which correspond to adding “1” and “0” respectively in the tallying process);
2. *Concealing.* Given only one cryptogram in a selected row, it is not possible to distinguish whether it is “Yes” or “No”;
3. *Revealing.* Given both cryptograms in a selected row, anyone is able to tell which one is “Yes” and which is “No”;
4. *Self-tallying.* Given an arbitrary selection of a cryptogram from each row, anyone is able to tally how many “Yes” votes there are in the selection.

The protocol allows the flexible choice of pre-computation in the implementation. In the description above, all cryptograms are pre-computed before the election, which serves to minimize any latency in voting on election day. Alternatively, the cryptograms can be computed on-demand in real time during voting. If cryptograms are pre-computed, they will need to be kept confidential alongside the secret x_i values; in contrast, if cryptograms are computed on-demand, only the x_i values need to be kept secret.

Table 13.2: Setup phase.

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

Note: Data in the first three columns are published on a public bulletin board before the election as commitment so they cannot be retrospectively changed later. Data in the last two columns are kept secret; they are either computed on-demand during voting or pre-computed before the election.

13.4.3.2 Voting

On election day, voters enter the polling station with their ID documents for authentication. After being authenticated successfully, each voter randomly takes an authentication token, which may be a smart card or a one-time passcode. The voter then enters a private booth and uses the token to authenticate herself to the voting machine. The token allows the voter to cast one vote. The voter's real identity remains unknown to the machine.

On the touchscreen interface, the voter is promoted to select a choice for the single-candidate election: “Yes” or “No”. To cast the vote, the voter follows two basic steps.

In the first step, she touches the screen to select a choice. The DRE machine prints the following data on the paper receipt: the ballot serial number i and the cryptogram of the selected choice (see Figure 13.2). The receipt is appended with a digital signature signed by the machine to prove the authenticity of the printed data.

In the second step, the voter is given the option to either confirm or cancel the previous selection (Figure 13.2). If the voter chooses “confirm,” the machine continues to print “finish” on the same receipt, followed by a digital signature that covers the entire receipt. In this case, a valid ballot has been cast. If the voter selects “cancel,” the machine prints the selected choice in plaintext (i.e., “Yes” or “No” in this single-candidate example), followed by the other (unused) cryptogram and a digital signature. In this case, a dummy vote has been cast. The voting interface is then returned to the initial Yes/No screen for the voter to start over with a different (unused) ballot. As will become clear later, a dummy vote does not add to the total tally. The voter is free to cast as many dummy votes as the system allows (up to the safety factor defined in the setup stage), but is restricted to cast only one confirmed vote.

The option of “cancel” is to allow user auditing. When the voter makes a choice, a single cryptogram is printed on the receipt. Based on the “concealing” property, a single cryptogram does not reveal any information about the user's choice. This is necessary for preventing coercion and vote selling. However, a dishonest DRE ma-

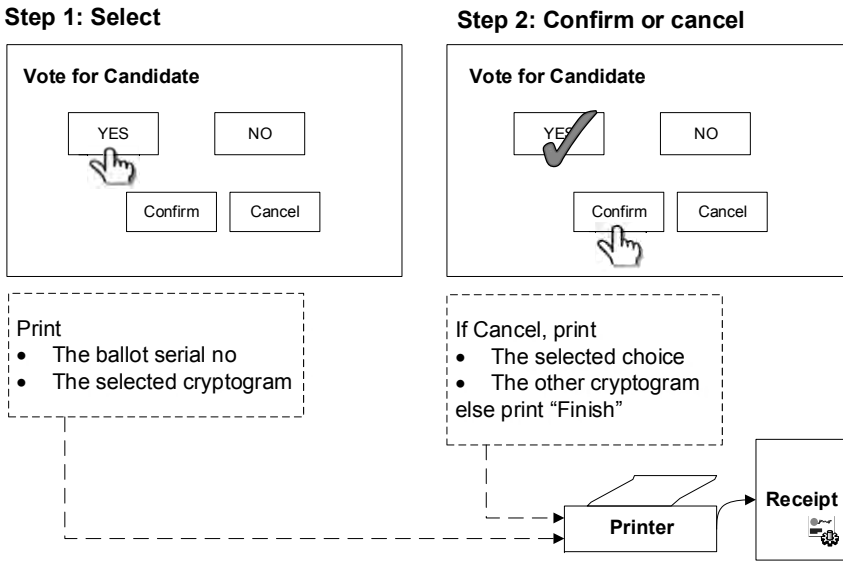


Figure 13.2: The voting procedure in the DRE-i protocol.

chine may cheat by swapping the cryptograms, so the cryptogram of the unselected choice is printed on the receipt. This swapping-attack will be detected once the voter chooses to cancel the vote based on the “revealing” property. At this point, the voter checks if the plain-text choice printed on the second part of the receipt matches her selection in the previous step. If not, she should raise a dispute immediately to the administrative staff in the polling station.

After voting, the voter leaves the private booth with a receipt for the confirmed ballot and possibly several receipts for cancelled (dummy) ballots. To ensure her vote has indeed been included in the tallying system, the voter checks that the same data on receipts (including dummy votes) are published on the bulletin board. This can be done with the assistance of voluntary helpers in the polling station. The data printed on the receipt is essentially public, so it can be shown to anyone. In a real election, this kind of verification is best performed on-site, say before exiting the polling station; then any irregularity can be identified and dealt with immediately.

When the election finishes, any unused ballots will be automatically cancelled by the system with the two cryptograms published on the bulletin board. These are effectively treated as dummy votes.

13.4.3.3 Tallying

When the election is finished, the DRE machine announces the tally of the “Yes” votes that it has counted internally, just like existing practice in a real-world DRE-

Table 13.3: Ballot tallying.

No i	Random pub key g^{x_i}	Restructured pub key g^{y_i}	Published Votes V_i	ZKPs
1	g^{x_1}	g^{y_1}	Valid: $g^{x_1 \cdot y_1}$	a 1-of-2 ZKP
2	g^{x_2}	g^{y_2}	Valid: $g^{x_2 \cdot y_2} \cdot g$	a 1-of-2 ZKP
3	g^{x_3}	g^{y_3}	Dummy: $g^{x_3 \cdot y_3}, g^{x_3 \cdot y_3} \cdot g$	two 1-of-2 ZKPs
...
n	g^{x_n}	g^{y_n}	Dummy: $g^{x_n \cdot y_n}, g^{x_n \cdot y_n} \cdot g$	two 1-of-2 ZKPs

based election. However, in contrast to existing practice, the DRE machine must publish additional audit data, which includes making all receipts available on the bulletin board. The audit data allow the public to verify the integrity of the announced tally.

Public verification simply involves multiplying all the published votes V_i (see Table 13.3 for an example). For dummy ballots, only the no-votes are included in the multiplication. Thus, we have:

$$\prod_i V_i = \prod_i g^{x_i y_i} g^{v_i} = \prod_i g^{v_i} = g^{\sum_i v_i}$$

Because of the cancellation of all random factors (based on the “cancellation formula” in Proposition 13.1), the result of the multiplication is $g^{\sum_i v_i}$, where the exponent represents the tally. Although the exponent can be calculated by exhaustive search, there is actually no need to do so in DRE-i. Because the machine announces the tally, one merely needs to verify if the tally is correct. This is done by raising the base g to the power of the announced tally and comparing it against $g^{\sum_i v_i}$. This operation takes only one modular exponentiation. Similarly, in a multi-candidate election, exhaustive search for the tallying results is not necessary either. Verifying the tally is much easier than calculating it.

13.4.4 Publication of the DRE-i Paper

When the design of the DRE-i protocol was complete, we made the paper publicly available at IACR ePrint (No. 452, 2010) [289], and meanwhile tried to find it a good home for its publication.

However, the difficulty of getting the DRE-i paper published exceeded my expectation. What the DRE-i protocol shows is a new way of constructing an E2E verifiable voting system – it differs from previous E2E designs in that it realizes the E2E verifiability *without* involving any tallying authorities. But, publishing a new idea in academic research is never easy. The paper was repeatedly rejected by various con-

ferences in the field. It was not until four years later that it was finally published by the *USENIX Journal of Election Technology and Systems* (Vol. 2, No. 3, 2014) [288].

Part of the reason for rejection is that the system was not implemented in any practical application, so its practical feasibility remained unclear to the reviewers. This gave me the motivation to build a DRE-i prototype and put theory to the practical test.

13.5 Trial Elections

In December 2010, after working in the security industry for three years, I joined the School of Computing Science, Newcastle University, as a lecturer. I was glad to become a full-time researcher again, and was able to find more time to pursue the DRE-i work further.

13.5.1 Prototyping DRE-i

Shortly after I joined the faculty, my research was jump-started by the support of an internal Research Initiative Fund (RIF) in the school. The funding allowed me to recruit a research assistant for six months to implement DRE-i and conduct trial elections. Although the funding only lasted for six months, the support was timely and encouraging, especially because the DRE-i paper was still going through the process of receiving rejections. As it happened, Dylan Clarke, a final year PhD student in the school, was looking for an assistant post. Clarke had several years professional software development experience before doing his PhD. He proved to be an ideal candidate for this project.

For the ease of conducting trials, we decided to have an Internet-based implementation of DRE-i, instead of a local touchscreen-based one. The protocol remains basically the same as described in Section 13.4.3. The pre-computation is the same as before. During voting, the voter follows the same two steps to cast or audit a vote. However, instead of using a touchscreen, the voter uses a web interface. The receipt is displayed on the web page, instead of being directly printed out on paper as in local voting. The tallying process is exactly the same as before.

While the underlying DRE-i protocol remains the same, using it for Internet-based voting instead of polling-station-based voting has security implications. First, since there is no physical private booth, a voter may now vote under the direct duress of a coercer. Second, as voting is remote, a voter may simply transfer (or sell) her voting credential to a third party. Third, if the voter finds any mismatch of data in the printed receipt, it is no longer possible to raise the dispute and have it be dealt with swiftly on the spot. For these reasons, Internet voting is commonly considered only suitable for elections with low coercion risks. This fits the nature of the trial elections that we were about to conduct.

In just two months, Clarke completed a prototype of the DRE-i voting system. We were ready to put the prototype to a practical test.

13.5.2 *Favorite Chocolate Election*

As for the first trial, our goal was to have something fun and involve as many participants as possible. Hence, we decided to conduct a “favorite chocolate” election.

The trial was held in October 2011. We bought three boxes of chocolates of different brands: *Quality Street*, *Roses* and *Celebrations*. The boxes were put on a table in a common room, where staff and postgraduate students usually had tea breaks during the day. Next to the chocolates was a box of randomly mixed paper slips. Each slip printed a web voting address and a one-time passcode. A signboard was put on the table, inviting people to taste different chocolates, take a random paper slip and vote for their favorite chocolate.

With the paper slip, a voluntary participant could use any computer to vote. After entering the passcode onto the voting website, the voter was shown a set of choices: voting for *Quality Street*, *Roses* or *Celebrations* (Figure 13.3(a)). Following the DRE-i protocol, voters voted for their favorite candidate. In the end, 39 people participated and cast their votes. The winner was *Quality Street*, which received 18 out of 39 votes. Besides displaying the voting results, the bulletin board (Figure 13.3(b)) also contained a link to download the audit data (an XML file) and an open-source Java program for verifying the audit data.

After voting, voters were asked to answer a questionnaire and also write free-text comment in the feedback. The questionnaire consisted of a set of six questions. For each question, voters were asked to indicate their agreement or disagreement on a Likert scale from 1 to 5 (i.e., “strongly agree,” “agree,” “neural,” “disagree” and “strongly disagree”). The questionnaire answers and free-text comments are summarized in Figure 13.3(c).

The user feedback was generally positive and encouraging. With minimum instruction (just a web address and a one-time passcode), voters generally understood how to vote and found voting easy.

However, this trial also exposed two main issues. The first was the low auditing rate. Only 4 out of 39 people (i.e., 10%) tried user auditing (by clicking the “cancel” option). The purpose of user-initiated auditing was not clear to many voters. The second issue concerns the difficulty of verifying the receipt. The receipt contained a long string of random-looking characters (in base-64 encoding). Comparing this random string with the one published on the bulletin board proved not that easy for a human.

You can choose to vote for the following candidates.

Candidate	Picture	Select
Quality Street		<input type="radio"/>
Roses		<input type="radio"/>
Celebrations		<input type="radio"/>

(a) Web voting interface

Election Results - Favorite Chocolate Election

Candidate	Votes
Quality Street	18
Celebrations	11
Roses	10

Ballot Type	Number
Confirmed	39
Cancelled	4
Unused	107

[Download the GUI Election Checker](#) - requires Java 5.0 or greater.

[Download the GUI Election Checker Source Code](#)

Right click to save the receipt XML file for verification

Ballot Number	Link
0	Click Here
1	Click Here
2	Click Here
3	Click Here

(b) Result on the web bulletin board

Feedback Page - Favorite Chocolate Election

The current feedback results are:

Statement	Average Score Given	Nearest Option to Average Score
I understood how to vote.	1.57	Agree
Voting was easy.	1.78	Agree
I understood how to check my ballot had been recorded correctly.	2.39	Agree
Checking that my ballot had been correctly recorded was easy.	3.13	Neutral
I understood why I was being asked to check ballots.	2.87	Neutral
I felt confident that my vote had been recorded correctly.	2.43	Agree

The comments received were:

A lot of these answers are biased - I have a good idea how the system works.

Checking the cryptograms was too long - I just looked at the first and last entry - might be easier ways of doing this? with an image perhaps?

Wrong https certificate gave an immediate bad impression. The cryptograms were so long as to be next to impossible to check by hand. The whole process would be completely opaque to someone without a Computing Science degree. The only electronic voting system I would trust is one which counts my vote electronically, and which I can see a ballot paper being retained for later hand scrutiny.

(c) Summary of questionnaire answers and free-text feedback

Figure 13.3: Favorite chocolate voting.

13.5.3 Favorite Cheese Election

Based on the user feedback, we improved the DRE-i prototype. Instead of using the 2048-bit multiplicative group in a finite field (DSA-like setting), we changed to implement DRE-i using the NIST-256 (ECDSA-like setting) additive group over an elliptic curve. This makes the representation of the cryptograms more compact. By reducing the size of the receipt, our hope was that voters would find it easier to

verify the receipt. When the changes were done, we decided to have a second trial. This time, we chose to conduct a “favorite cheese” election.

The second trial was held in November 2011. We bought three different brands of cheeses: *Wensleydale*, *Camembert* and *Blue Stilton*. As before, we put the cheeses (along with crackers) in the common room, and also a box of randomly mixed passcode slips. Staff and postgraduate students were invited to taste different cheeses, take a random passcode slip, and vote for their favorite cheese. Voluntary participants cast their votes through a web interface, as shown in Figure 13.4(a). This time, 35 people cast their votes. The winner was *Wensleydale*, which received 14 votes. The tallying results are shown in Figure 13.4(b). The answers to the questionnaire and free-text comments are presented in Figure 13.4(c).

While the general feedback from participants was still positive, we observed that the same two issues remained.



First, this time the auditing rate was even lower. Only 1 out of 35 people chose auditing (i.e., 3%), which was lower than the previous 10%. This was likely because many of the voters participated in the previous trial. When they were more familiar with the system, they became less inclined to verify it. However, if the auditing rate is very low, this can invalidate the basic assurance of “individual verifiability” in an E2E voting system. This shows that if voters are entirely left to their own devices to audit votes, it is likely very few of them will actually do it.

Two solutions may help address this problem. The first one was originally suggested by my colleague, Brian Randell, and we named it the “Waitrose scheme” [290]. The solution was inspired by an existing practice in Waitrose (which is a chain of British supermarkets). When customers shop in a Waitrose supermarket, they are given a charity token at the cashier. At the exit of the supermarket, customers are free to donate the token to a preferred charity organization by dropping it in the designated box. The similar idea can be applied to encourage auditing in e-voting. If voters choose to audit their vote, they receive a charity token, which they can donate to a preferred charity organization near the exit of the polling station. Randell vividly described the essence of this solution as “ethical bribery” (which is a term I particularly like). Another practical solution is to employ dedicated auditors from different voting parties. A dedicated auditor is allowed to vote at any time during the election day, but is limited to cast dummy votes only. In a real election, we expect that the two solutions might be combined to improve the user auditing rate.

Second, though the data on the receipt became more compact, it remained difficult for a voter to compare the receipt against the bulletin board. The main limiting factor is the size of the digital signature. Using ECDSA at the 128-bit security level, the size of a digital signature is at least 64 bytes, i.e., 85 characters using the base-64 encoding. Comparing two strings of 85 random characters is trivial for a computer, but can prove rather difficult for a human.

It seemed unavoidable that we needed to limit the size of a receipt to a very short human-readable string. Through experiments, we determined that a string of 10 characters was acceptable to common users. This was adopted in the subsequent

You can choose to vote for the following candidates.

Candidate	Picture	Select
Wensleydale		<input type="radio"/>
Camembert		<input type="radio"/>
Blue Stilton		<input type="radio"/>

(a) Vote for favorite cheese

Election Results - Favorite Cheese Election

Candidate	Votes
Wensleydale	14
Camembert	13
Blue Stilton	8

Ballot Type	Number
Confirmed	35
Cancelled	1
Unused	114

Download the GUI Election Checker - requires Java 5.0 or greater.

Download the GUI Election Checker Source Code

Right click to save the receipt XML file for verification

Ballot Number	Link
0	Click Here
1	Click Here
2	Click Here
3	Click Here

(b) Vote for favorite cheese

Feedback Page - Favorite Cheese Election

The current feedback results are:

Statement	Average Score Given	Nearest Option to Average Score
I understood how to vote.	1.43	Strongly Agree
Voting was easy.	1.81	Agree
I understood how to check my ballot had been recorded correctly.	2.43	Agree
Checking that my ballot had been correctly recorded was easy.	3.10	Neutral
I understood why I was being asked to check ballots.	2.76	Neutral
I felt confident that my vote had been recorded correctly.	2.52	Neutral
I felt confident that my vote was anonymous.	2.43	Agree

The comments received were:

I had the problem with "l" vs "1" suggest use of different font on passcodes

Passcode quite long & fiddly but I guess this is needed.

Next time bring beer to sample!

Pre-selecting your preferred candidate and "confirm" makes it far too easy to elect a dictatorial Tyke.

(c) Vote for favorite chocolate

Figure 13.4: Favorite cheese voting.

prototyping of a verifiable classroom voting system based on DRE-i. The shortening of the receipt presents a trade-off between security and usability. While it significantly improves usability in verifying the receipt, the fact that the string is too short to contain a digital signature makes it subject to false claims (digital signatures are still available on the bulletin board, but not printed on receipts). A dishonest user may modify the receipt and claim it mismatches the data published on the bulletin board. It will not be easy for a third party to distinguish if the system prints the wrong

receipt or the user makes a false claim (in order to discredit the voting system). In practice when such a dispute arises, it needs to be dealt with on a case-by-case basis.

Besides the two issues, another important lesson we learned from this trial is that details really matter. The result that *Wensleydale* was the winner was most likely helped by the fact that it was a pre-selected choice in the voting interface. The same observation applies to explain why *Quality Street* won the previous trial (since it was also a pre-selected choice; see Figure 13.3(b)). So, this seemingly innocuous detail in the implementation may have inadvertently biased the voting outcome. We did not realize this issue until the result of the second trial was available and then the correlation between the winner and the pre-selected choice became evident. (This issue was also pointed out by an anonymous comment; see Figure 13.4). In another example, the passcode was initially generated as a random mixing of upper, lower letters, digits and symbols to get the maximum entropy. However, voters found it difficult to distinguish the digit “1” from the capital letter “I” in the printed passcode. They also found it difficult to enter capital letters and symbols on a small-screen mobile phone. This prompted us to change to use Crockford’s Base32 encoding (<http://www.crockford.com/wrmg/base32.html>) in the subsequent developments. These practical issues could have been easily neglected if the system had not been implemented and trialed in practice.

13.5.4 ERC Starting Grant on Self-Enforcing E-Voting

Despite the identified issues, the two trial elections greatly increased our confidence about the practical feasibility of DRE-i. Because of the removal of TAs, managing an election became almost effortless. The setup was done in minutes. Once the setup was completed, the election could commence immediately. Voters did not need to perform any cryptographic operation in the client browser, hence they could just use a plain web browser (without enabling any Java plug-in or JavaScript) on a slow computer (say a mobile phone) without noticing any significant latency in voting. Finally, the voting results were instantly available once the election was ended, since there was no delay in waiting for TAs’ inputs.

The success of the preliminary trials highlighted the potential of a new category of voting systems that are E2E verifiable and TA-free. We called this new category “self-enforcing e-voting” (SEEV). The “self-enforcing” property for security protocols is generally considered powerful, since it serves to minimize the dependence on trusted third parties. However, the concept of “self-enforcing e-voting” had not been actively explored by previous research. The work of DRE-i shows that a “self-enforcing e-voting” system is possible for simple approval type voting schemes, but the question remains if DRE-i can be extended to support more complex ranking-based voting schemes such as single transferable vote (STV) (see Chapter 4 for details on STV).

Towards the end of 2011, I submitted a research proposal to the European Research Council (ERC) to continue the investigation on SEEV. About six months

later, I was informed that my application was successful – ERC agreed to provide a five-year Starting Grant of €1.5m (2013-2017). This grant allowed me to assemble an interdisciplinary research team to investigate SEEV further.

In 2013, two researchers, Siamak Shahandashti and Peter Lee, joined my ERC team, supported by the ERC Starting Grant. They soon proposed an improvement to the basic DRE-i scheme. In DRE-i, the verification of the tallying integrity involves multiplying *all* the encrypted votes published on the bulletin board. If one or more votes are missing (or corrupted), the verification process will fail. This is a potential weakness that worried some reviewers. To address this dependability issue, Shahandashti and Lee came up with an efficient fail-safe mechanism for DRE-i: i.e., in case one or more ballots are missing, the DRE system can gracefully recover from the effect of missing ballots by publishing additional audit data that allow the integrity of the remaining votes to be verified while preserving the secrecy of the missing ballots (details can be found in [288]). This fail-safe solution is a significant enhancement to the theory of the basic DRE-i protocol, and contributes to the final acceptance of the DRE-i paper by the *USENIX Journal of Election Technology and Systems* in 2014 [288]

13.5.5 A Verifiable Classroom Voting System

The two trial elections proved to be an invaluable experience to us. The desire to run more e-voting trials directed us to “classroom voting.”

Classroom voting is a modern pedagogy, invented by Professor Eric Mazur in the 1990s during his teaching of physics at Harvard University, and later extended to teaching mathematics, chemistry and other subjects [385]. Newcastle University, among several other UK universities, has been trialling this pedagogy by using a commercial classroom voting product from the TurningPoint company.

As part of the training required for new academic staff in the UK, I attended a seminar on “changing how we teach with voting technology.” An instructor from Newcastle University Teaching Unit showed us a demo of the TurningPoint voting system. After a short briefing, he handed out a set of hand-held devices, known as “clickers.” Each clicker was able to communicate with a special receiver installed on the instructor’s computer through a radio-frequency wireless signal. To synchronize each clicker with the receiver, we were instructed to enter a 2-digital channel code on the clicker. Once the sync was completed, voting was ready to start. The instructor showed a multiple-choice question over a PowerPoint slide. We were then asked to use the clicker to vote for the best answer. Each clicker had a small light indicator. A green light indicated that the vote had been recorded by the receiver, while an amber light indicated failure. Once everyone had voted for answers, the instructor closed the voting session by using TurningPoint software that was installed in PowerPoint as a plug-in. The tallying result of the answers was then displayed over the PowerPoint projector in a colorful bar chart. In a real class, the tallying result would give the



Figure 13.5: TurningPoint voting system. It comprises a receiver that is plugged into the USB port of a computer and a set of custom-built voting devices known as clickers.

teacher instant feedback about the students’ learning outcome, so the teacher could adjust teaching accordingly.

I was impressed by the pedagogical value of classroom voting, but as a security researcher, I could not help wondering if security aspects of voting had been considered. So I asked the instructor how could voters verify if the tallying results were accurate. The instructor replied: “You need to trust the system.” He went on to explain that one could gain the trust by testing the product before the class.

The instructor’s reply echoed similar arguments in other e-voting applications. For example, many e-voting systems deployed in real-world national elections were unverifiable. Voters cast their votes using electronic means (e.g., through a DRE machine) but were not able to verify if the votes had been recorded and tallied correctly. When the security concern of such voting systems was raised, the typical response from the government was: “You need to trust the system.” Usually, the “trust” is formalized by a process of certification: a panel of authorities examine the product and declare it “trustworthy” since nothing wrong is identified. However, without verifiability, if there is anything wrong with the tallying (e.g., due to software bugs, implementation error or malicious tampering), voters will not be able to notice.

In a classroom voting application, one might question if the integrity of the voting outcome is worth protecting. After all, if the tallying results turn out to be wrong, it seems to do no serious harm. However, I believed the “integrity” was still important in a classroom voting application. While questions in classroom voting are

typically insensitive, sometimes they do carry stakes. For example, students may be asked to assess the module or rate the performance of the lecturer. In those cases, the integrity of the voting process should be guaranteed and the results be externally auditable (e.g., by university administrative units). Adding “verifiability” to the classroom voting process can significantly broaden the traditional scope of classroom voting, covering all types of voting questions from low to high stakes.

From a research perspective, classroom voting presents an ideal opportunity to put the theory of verifiable voting to practical test. If a verifiable e-voting scheme is really practical, one should be able to apply it to practical voting scenarios on a *routine* basis without incurring any significant cost in usability. It is important that students should find using a “verifiable” e-voting product as convenient as using an “unverifiable” e-voting counterpart. If the “verifiability” is obtained at the great expense of convenience, voters would most likely vote with their feet, refusing to use the product eventually.

Based on the above considerations, we decided that it was worthwhile to implement a verifiable classroom voting (VCV) system based on DRE-i and try it out. I submitted a proposal to Newcastle University Innovation Funds about developing such a VCV system for pedagogical use. My proposal was subsequently accepted by the university teaching unit with a £6,000 pump-priming fund to support the system development. This enabled me to recruit two students to work on the project: Carlton Shepherd who was a second year Computer Science undergraduate, and Dylan Clarke who was then a fresh PhD graduate.

After three months’ hard work during the summer of 2012, Shepherd and Clarke completed a working prototype of the VCV system. The back-end is a web server hosted at the School of Computing Science, Newcastle University. All cryptographic operations are performed at the server side. On the front-end, the prototype supports three different voting interfaces: a web browser, an Android app and an iPhone app. Among the three, the web browser is generally recommended, as it does not require installing any app. However, some students prefer apps to a web browser, so the Android/iPhone apps are still provided. Figure 13.6 shows the web voting interface, which also contains links at the bottom to 1) help; 2) the Android app; 3) the iPhone app and 4) coordinator’s login. Any user with a Newcastle University campus account is able to log in as a coordinator and create classroom voting sessions.

The VCV system supports two types of authentication: using group passcode and individual passcode. The former uses a single passcode for the whole voting session, which is the most common choice in the routine use of the system. A teacher creates a voting session with a set of questions and answers, and specifies a single passcode (which may be empty). During the class, all students are allowed to vote after entering the passcode. Of course, a student may reuse the passcode to vote more than once. In most scenarios, this is an acceptable trade-off, given the low-stake nature of the voting result. In other scenarios where voting results carry high stakes, the one-man-one-vote rule must be strictly enforced. In this case, the second authentication option should be used. The system generates a list of random passcodes that match



Figure 13.6: Web interface for classroom voting.

the estimated number of legitimate voters. The passcodes are printed on physical paper and cut into small paper slips with each slip containing one random passcode. All paper slips are physically mixed in a container before being manually distributed to voters. (This method was used in the 2015 “best paper” voting competition in our school in January 2015.)

13.5.6 *Cryptography Meeting Pedagogy*

Since the first semester in October 2012, I have been using the VCV system in real teaching in my own classes: i.e., a “System Security” module for MSc students and a “Cryptography” module for BSc students in the School of Computing Science. Normally I create a voting session on the day before the lecture. On the lecture day, I use the first ten minutes for voting, asking students to select best answers for a set of multi-choice questions that are drawn from the previous lecture. To make the voting process more interactive, I encourage students to discuss with peers who sit next to them before voting, to ensure they have seriously thought about answers before sending votes. Once students have answered all recap questions, I close the election by using a web interface (after logging on as coordinator using the Newcastle University campus account). The tallying results are instantly shown on the bulletin board (a web page). Based on the results, I can quickly assess the student learning outcome and adjust my teaching accordingly.

By practicing verifiable classroom voting in real teaching on a routine daily basis, I came to appreciate the powerful potential of this modern pedagogy. The use of

Table 13.4: Student survey.

Q1: Does the voting make the lecture more fun?	
<p>1. Yes: 28 2. No: 4</p> <p>A horizontal bar chart with two bars. The first bar (labeled '1') is dark grey and extends to 88%. The second bar (labeled '2') is light grey and extends to 13%.</p>	<p>1. Yes: 20 2. No: 3</p> <p>A horizontal bar chart with two bars. The first bar (labeled '1') is dark grey and extends to 87%. The second bar (labeled '2') is light grey and extends to 13%.</p>
Q2: Does the voting help you learn?	
<p>1. Yes: 31 2. No: 2</p> <p>A horizontal bar chart with two bars. The first bar (labeled '1') is dark grey and extends to 94%. The second bar (labeled '2') is light grey and extends to 6%.</p>	<p>1. Yes: 21 2. No: 2</p> <p>A horizontal bar chart with two bars. The first bar (labeled '1') is dark grey and extends to 91%. The second bar (labeled '2') is light grey and extends to 9%.</p>
Q3: Do you find it useful to have a small group discussion before voting?	
<p>1. Yes: 23 2. No: 8</p> <p>A horizontal bar chart with two bars. The first bar (labeled '1') is dark grey and extends to 74%. The second bar (labeled '2') is light grey and extends to 26%.</p>	<p>1. Yes: 19 2. No: 4</p> <p>A horizontal bar chart with two bars. The first bar (labeled '1') is dark grey and extends to 83%. The second bar (labeled '2') is light grey and extends to 17%.</p>
Q4: What do you think of the anonymity of voting?	
<p>1. I like the voting being anonymous: 21 2. I prefer the voting not to be anonymous: 3 3. I don't mind either way: 7</p> <p>A horizontal bar chart with three bars. The first bar (labeled '1') is dark grey and extends to 66%. The second bar (labeled '2') is light grey and extends to 10%. The third bar (labeled '3') is black and extends to 23%.</p>	<p>1. I like the voting being anonymous: 17 2. I prefer the voting not to be anonymous: 0 3. I don't mind either way: 6</p> <p>A horizontal bar chart with three bars. The first bar (labeled '1') is dark grey and extends to 74%. The second bar (labeled '2') is light grey and extends to 0%. The third bar (labeled '3') is black and extends to 26%.</p>
Q5: Do you recommend classroom voting for teaching the same module next year?	
<p>1. Yes: 30 2. No: 1</p> <p>A horizontal bar chart with two bars. The first bar (labeled '1') is dark grey and extends to 97%. The second bar (labeled '2') is light grey and extends to 3%.</p>	<p>1. Yes: 21 2. No: 2</p> <p>A horizontal bar chart with two bars. The first bar (labeled '1') is dark grey and extends to 91%. The second bar (labeled '2') is light grey and extends to 9%.</p>

Note: The results on the left were from a 2014/2015 BSc class in Cryptography, and on the right from a 2014/2015 MSc class in System security.

classroom voting has evidently improved the student interactions in the class and made them more engaged. To gauge the effectiveness of using VCV for pedagogy, in January 2015, I conducted two student surveys in the “System Security” (MSc) and “Cryptography” (BSc) classes respectively. Students were asked to select answers to five questions. The answers are summarized in Table 13.4.

As shown in Table 13.4, the vast majority of students (around 90%) agreed that the classroom voting system made the lecture more fun and helped them learn. Between 74% and 83% students found the small group discussion useful. While only a few students did not mind whether voting was anonymous, the majority (68-74%) liked the current design that preserved anonymity. Finally, nearly all students (91-97%) supported the continued use of classroom voting in teaching in the following year. These results are largely consistent with the previous surveys conducted in [290] and [287].

Since the VCV system became first available for practical use in a live teaching environment in 2013, it has been used by students at both the undergraduate and postgraduate levels. Based on students' feedback, the system was improved and made freely available across the campus to anyone who has a valid Newcastle University account. Gradually, the system has been adopted by lecturers across the university, including computing science, electrical engineering, politics and business.

The public verifiability of the VCV system is a distinguishing advantage over all other classroom voting systems. Voters can vote anywhere, at any time with ability to independently verify the integrity of the tallying results. This makes it possible to extend the VCV system beyond the traditional classroom voting. For example, in 2015 the VCV system was used by members of the School of Computing Science at Newcastle University to vote for the "best paper" that had been published in the school during 2014. By comparison, this kind of election cannot be securely supported by existing commercial classroom voting systems due to a lack of verifiability.

In future, we plan to extend the VCV system for public use outside Newcastle University. Our aim is to make the system as freely available as possible over the Internet, so universities around the world can benefit from practicing the modern classroom voting pedagogy. Our business plan has recently been approved by the 2015 ERC Proof-of-Concept grant (which supports commercialization of ERC research results). Work is currently in progress in this regard.

13.6 Conclusion

This chapter describes a ten-year journey of exploring a practically useful e-voting system. The journey started from a seeming coincidence – the discovery of an efficient solution to the Dining Cryptographers problem. The essence of that solution, through the cancellation of random factors by juggling public keys, was subsequently applied to construct efficient e-voting protocols for both small-scale boardroom voting and large-scale voting. The end result of this journey is a new E2E verifiable voting system called DRE-i. As compared with other E2E voting systems, DRE-i does not require any tallying authorities, and hence the system is "self-enforcing." Trial elections based on the DRE-i protocol have been conducted with promising results. A verifiable classroom voting system, based on DRE-i, is currently being used

at Newcastle University for pedagogical purposes, and will be publicly available in the near future.

In retrospect, one interesting observation of this journey is that it is fueled by uncertainty. At each stage of the journey, the goal towards the next challenge is clear, but its reachability has remained largely uncertain until a solution is finally worked out. The next challenge that we are working on is how to extend DRE-i to support more complex voting schemes, such as STV. I have no answer at this stage and I do not even know if this is achievable – that feelings of *déjà vu* remind me that the journey is still continuing.

Acknowledgment

The author would like to thank Brian Randell for his invaluable comments and feedback. This work is supported by the ERC Starting Grant, No. 306994.