uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Aleksander Essex

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Punchscan: Designing an Independent Verification Mechanism for Elections

TITRE DE LA THÈSE / TITLE OF THESIS

Professor C. Adams

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Professor G-V. Jourdan

Professor G. Yee

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Punchscan: Designing an Independent Verification Mechanism for Elections

Aleksander Essex

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in Electrical Engineering

School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-41660-0
Our file Notre référence
ISBN: 978-0-494-41660-0

Abstract

Punchscan is an open-source vote-counting system, the results of which are independently verifiable by voters through their participation in the election audit process. Voluntary and universally available, this audit process establishes an overwhelmingly high statistical degree of confidence in the integrity of the outcome. At the same time it was developed around the recognition that the secret-ballot, and in turn voter privacy, remain a fundamental requirement in modern democracies. Punchscan offers voters the ability to "see their vote count," while at the same time protecting against improper influence.

This thesis describes the design and development of Punchscan from an information security and system design perspective. It begins by examining the concerns over contemporary electronic vote-counting systems, and introduces principles (such as independent verification) from which a more secure system can be built. We go on to explore the functional components, their design purpose, and the election/verification procedures of Punchscan. We present a case study of Punchscan's first use in a binding election. Finally we end with a discussion of how Punchscan met its design goals, and propose directions for future work.

Acknowledgements

Thanks foremost to the Punchscan project members – David Chaum, Richard T. Carback III, Stefan Popoveniuc and Jeremy Clark. We've collaborated as a team in the design, development and deployment of Punchscan. Thanks of course to Carlisle Adams – the best advisor a graduate student could have. Finally I wish to acknowledge the University of Ottawa's Graduate Students' Association (GSAÉD) who nobly took a risk on a new technology to help advance the research in their community.

Authors note Portions of chapter 6 and 7 are reprinted from [27] with permission.

The "voting" people know this well. For the rest:

"Those who cast the votes decide nothing.

Those who count the votes decide everything."

-Joseph Stalin

Forward

Electronic vote counting systems, a technology unknown to many Canadians, has made for an interesting challenge when attempting to put our research into context. When asked in polite conversation I'd simply answer that "we are trying to help the Americans with their voting problems;" an oversimplification that often garnered a degree of comprehension (and sympathy). Though our work has focused so far on what's primarily been an "American problem", the ultimate realization of this field of research carries the potential to strengthen any democratic process in use today–even Canada's.

Originally conceived by David Chaum in late 2005, I joined the Punchscan team in May 2006. We ran the first mock election in August 2006 in Canada, the U.S., and Europe. In November 2006 we met at the National Press Club in Washington D.C. to announce the (open-source) release of the Punchscan 1.0 software implementation. In March 2007, we organized and ran the first ever E2E voting system successfully used in a binding election, right here at the University of Ottawa. In July 2007 we competed in the VoComp student voting system design competition in Portland Oregon and won the $10000 grand prize. Punchscan has appeared in the IEEE Spectrum, BBC World's Digital Planet, Slashdot, the Washington Post, CBC Radio's As it Happens, and will be mentioned in an upcoming American documentary film.

It's a promising start to a future in which electronic vote-counting systems do not "decide everything." Though I loathe to say it, Stalin was right. I only qualify it to say: this young field of end-to-end verifiable electronic vote-counting system research is in pursuit of a society in which "those who cast the votes" and "those who count the votes" are the same.

Aleksander Essex
Ottawa, Canada
December 2007

Contents

Forward i

1 Introduction 1

1.1 Organization and Contributions of this Thesis 2

2 Background 4

2.1 The Voting Problem 4

2.1.1 Integrity in Voting 5

2.1.2 Privacy in Voting 8

2.2 Survey of Vote-counting Systems 11

2.2.1 Australian Ballot 12

2.2.2 Direct Recording Mechanical 13

2.2.3 Optical Scan 13

2.2.4 Direct Recording Electronic 14

2.3 Independent Verification 14

2.4 E2E: End-to-End Cryptographic Independent Verification 16

2.5 Survey of E2E Systems 17

2.5.1 Prêt à Voter 18

2.5.2 Scratch and Vote 20

2.5.3 ThreeBallot 21

3 Architecture 23

3.1 Architectural Overview 23

3.2 Voting Components 24
3.2.1 The Punchscan Ballot 26
3.2.2 Polling Place 28
3.3 Election Administration Components 29
3.3.1 The Punchboard – Specification and Design Rationale 29
3.3.2 Diskless Workstation 42
3.3.3 Trusted Printer 43
3.4 Independent Verification Components 44
3.4.1 Pre-Election Audit 44
3.4.2 Post-Election Audit 45
3.4.3 Receipt Checker 46
3.5 Audit Challenge Generator 46

4 Functional Specifications **49**

4.1 Ballot Template Design 49
4.2 Ballot Authoring Tool 50
4.3 Election Engine 52
4.3.1 Software Validation Tool 52
4.3.2 Election Master Key Creation 53
4.4 Polling Place Software 56
4.5 Cryptographic Specification 57
4.5.1 Ballot Keys 57
4.5.2 Commitments 58
4.5.3 Punchboard Permutations 59
4.6 Audit Data Specification 59
4.6.1 Definition of Terms 60
4.6.2 Data Files Used and Produced by the Meetings of the Trustees . . . 61
4.7 Election Audit Specification 63
4.7.1 Audit Challenge Generators 64

4.7.2 Pre-Election Audit Tool 66
4.7.3 Post-Election Audit Tool 66
4.7.4 Online Receipt Checker 67

5 Election Procedures **71**

5.1 Trustee Procedure 73
5.1.1 Election Definition Phase 73
5.1.2 Pre-Election Phase 74
5.1.3 Election Phase 74
5.1.4 Post-Election Phase 74
5.2 Voter Procedure 75
5.3 Poll Worker Procedure 76
5.4 Independent Verifier Procedure 77

6 Campus Election Results **79**

6.1 Election Requirements 80
6.1.1 Ballot Receipt Digital Signatures 80
6.1.2 Paper-based Backup 81
6.1.3 Electronic Pollbook 82
6.1.4 Ballot Clipboard and Lock 83
6.2 Preparing for the Election 84
6.2.1 Ballot Manufacture 85
6.2.2 The First Meeting of the Trustees 85
6.2.3 Pre-Election Audit 86
6.2.4 Printing Ballots 86
6.2.5 Poll Worker Training 87
6.3 Conducting the Election 87
6.3.1 Contests 87
6.3.2 Polling Station 88

6.4 After the Election 88
6.4.1 Election Results 88
6.4.2 Online Receipt Check 88
6.4.3 Post Election Audit 89
6.5 Incidents and Assistance Requests 89
6.5.1 Technical Issues 89
6.5.2 Psychological Acceptability 90
6.6 Metrics 94
6.6.1 Average time to vote 94
6.6.2 Cost 94
6.7 System Performance 95

7 Discussion and Conclusion **96**
7.1 Summary of Security Design 96
7.1.1 Integrity Protection Measures in Punchscan 96
7.1.2 Privacy Protection Measures in Punchscan 97
7.2 Punchscan in the scope of E2E 98
7.3 Design Comparison 101
7.4 Future Work 104
7.4.1 Usability Testing 104
7.4.2 Formal Design Criteria and Standards Compliance 105
7.5 Conclusion 106

A Glossary of Acronyms **115**

Chapter 1

Introduction

Think back to the last time you voted in an election. As you cast your ballot, did you stop to wonder, "will my vote *really* be counted?" If you are Canadian, chances are you did not. Indeed a recent Elections Canada study [49] found that, of the reasons for not voting, problems with election administration was of least concern (1% of all respondants). Conversely in the United States concerns over the comparatively widespread use of *electronic vote-counting machines* has sparked ongoing nation wide concern over its ability to produce election results free from error or improper influence. Recent studies suggest that between 10-12% of American voters are not confident their vote will be counted as intended [3, 50], with as many as 83% confident that the use of electronic-based vote counting systems increases the potential for fraud [2].

The controversy has sparked government legislative initiatives such as the Help America Vote Act (HAVA) of 2002 [32], and most recently the Voter Confidence and Increased Accessibility Act of 2007 (H.R. 811) [62]. HAVA helped create the Election Assistance Commission (EAC) which, with the assistance of the National Institute of Standards and Technology (NIST), offers a Voluntary Voting Systems Guideline (VVSG). Since states administer and procure their own voting equipment, the VVSG provides state-level elections bodies common testing criteria to evaluate their equipment against. In a major move in August 2007, the California Secretary of State (Debra Bowen) decertified voting machines of three leading electronic voting equipment vendors after conducting a "top to bottom

reveiew," [9]. The decision was based primarily on testing conducted by researchers from the University of California in an attempt to "compromise the accuracy, security, and integrity of the voting systems without making assumptions about compensating controls or procedural mitigation measures that vendors, the Secretary of State, or individual counties may have adopted," [4]. The team concluded that under these conditions the security of all three systems could be compromised.

Though the problems inherent to the use of electronic voting machines were proposed as early as 1975 [59], solutions such as the *voter-verified paper audit trails* (VVPAT) have been, in relative terms, a more recent development [44]. VVPAT is a paper record (which in many implementations resembles a cash register receipt tape) that transcribes votes in a direct "human readable" format. It is intended to be a physical, independent, observable and parallel record to that of the electronic vote-counting machine's (comparatively unobservable) electronic memory. Though the idea has been in literature for 15 years, and despite the advances since, VVPAT constitute the primary present-day focus of the EAC and H.R. 811's attempts to enact a solution. However the effectiveness of VVPAT has been drawing criticism from a growing number of prominent voting analysts. Notable studies have been conducted by Dave Dill of Stanford [20] and Michael Shamos of Carnegie Mellon [61] into security vulnerabilities, as well as John Whack of NIST [63] investigation into privacy. Certainly from the perspective of contemporary information security design, physical mechanisms would not be the first choice for the protection of information systems. This thesis takes one of the first steps past physical security in elections by proposing *Punchscan*, a protection mechanism built on cryptographic (i.e., information) security principles.

1.1 Organization and Contributions of this Thesis

We begin in **chapter 2** by defining the voting problem from an information security-based system perspective. We then survey common vote-counting systems and identify their shortcomings with respect to the problem definition. From there we introduce the notion of independent verification and E2E and survey some notable examples. In **Chapter 3** we

discuss the architecture of Punchscan and its system components with a focus on design rationale with particular emphasis on the underlying cryptographic structure. In **chapter 4** we provide a functional description of the software components with an algorithmic sketch of the independent verification aspects. **Chapter 5** offers an election timeline and list of procedures for the various entities participating in the election. The principles of the first 5 chapters are applied in **chapter 6** via a case study of the Punchscan implementation as its used in a live election. Finally in **chapter 7** we offer a discussion and comparison with respect to our original design goals, as well as offer a direction for future work and conclude this thesis.

Punchscan was originally proposed by David Chaum and a basic description was originally presented by Popoveniuc and Hosp in [51]. The primary contribution of this author by this thesis has been to reverse engineer and vastly expand upon the initial description to provide **the first comprehensive document detailing Punschan's design, rationale, procedure and practice**. Specific contributions include:

1. A proposal for integrity and privacy design criteria for vote-counting systems.
2. Development of a formal specification and design rationale of key system components including the Punchscan ballot and cryptographic back-end (i.e., "Punchboard").
3. Development of a functional specification of key system processes including a complete listing of independent verification (i.e., audit) procedures and algorithms as well as user-based election procedures, roles, and timeline.
4. Complete survey of system administration, operation and verification components including trusted workstation, election software, ballot authoring and template tools.
5. A proposal for an audit challenge generator that uses stock indices.
6. Case-study of the first successful use of an E2E vote-counting system in a real-world (i.e., binding) election with discussion of new components and analysis of results pointing to directions of future work.

Chapter 2

Background

2.1 The Voting Problem

The voting problem is defined here from the perspective of information security and can be broken into two areas of study. Integrity ensures voter intent is correctly captured and aggregated free from error or undue influence exerted on the election administration. Likewise privacy ensures the confidentiality of voter intent to avoid undue influence exerted upon the voters. Like most fields of design there is often a tradeoff between the two. In an extreme case removing the constraint of privacy would allow for perfect integrity. A "show of hands" is an example where the correctness of the results is trivially verifiable; each vote can be directly attributed to a voter. This example carries the property of non-repudiation, meaning that the voter cannot later deny their choice. However in the case of secret-ballot elections, actual voting intent must remain unknown to all except the voter, and therefore the voter must be afforded protection to allow them to plausibly deny their actual choice. With no way to attribute votes to actual voters, it is easy to see the integrity property is confounded when attempting to design a secure system.

2.1.1 Integrity in Voting

The definition of integrity in this thesis as it applies to the voting problem is the ability to detect changes made to election data caused by error or fraud[1]. In traditional elections the "ability to detect" was only available to election administrators or special observers. In E2E elections we are attempting to provide this ability directly to voters as well.

The study of integrity can be applied (perhaps trivially) to many examples of human-technology interactions. Consider the operations that must be correctly performed when attempting to call someone on the telephone. Dialed as intended: given a phone number, you can figure out how to press the buttons to dial it? Signaled as dialed: does the telephone unit emit the DTMF tones (or pulses) consistent with the buttons you pressed? Routed as signaled: does the phone service provider correctly route your call based on the signal it received? If so, putting it together means your call gets *routed as intended.* Obviously the integrity chain is easy for the caller to validate if the correct person picks up the phone at the other end. But what about when its not?

In the circumstances where the call is not completed, one could have accidentally dialed the wrong number (i.e. not dialed as intended). One could've pressed the button too quickly for the DTMF tone to be registered by the telephone exchange (i.e. not signaled as dialed). Or as sometimes is the case in transatlantic calls (being routed through satellite and subject to reliability issues) the call is not routed as signaled. We can take these ideas and apply them to another technology: electronic vote-counting systems.

Similar to our telephone example, the chain of election integrity can be reduced to three properties that must be satisfied to ensure a proper election outcome. As an example consider a traditional secret paper-ballot election. The voter must be able to successfully mark their choice on the ballot. When the ballot is cast, the marks must be correctly recorded. In the case of a paper-ballot, this simply means placing the ballot in the ballot box. Finally the election officials must be able to correctly count each ballot as it was

[1]Integrity here is congruent with the standard definition in information security meaning the ability to *detect* unauthorized changes to data, as opposed to other fields where it means the inability to change, or even the ability to correct data.

recorded. These three properties will be referred to as:

- **Marked as Intended**,
- **Recorded as Marked**,
- **Counted as Recorded**.

If we can prove each property was satisfied, then we can prove the composition of the properties yield correct results, i.e., the election was *counted as intended*. The language of this definition has been modified from previous ones. Here we seek to more accurately reflect system-level function. Previous descriptions have used terms such as "cast as intended" and "counted as cast," [55]. For the purposes of this thesis, we regard the act of "casting" a ballot as a legal action and not a technical or functional one. In practice the legal act of casting a ballot is usually simultaneous with the technical act of vote recording. Since our focus here is on technology (especially information security), "marking" and "recording" will take the place of "casting" when considering system design and security aspects.

Marked as Intended

Marked as intended refers to the ability (or probability) that the voter is able to correctly input their voting intent into the framework of the intent capturing (recording) technique employed at the poll. The responsibility of this property is twofold. Firstly the voter has an onus to attempt to understand and accurately perform the marking procedure in order to correctly reflect their voting intent. Secondly the marking procedure and technology employed for the purpose of intent capture must be made usable and available to the entire range of needs of the voting population. Perhaps the most well known example of a failure of the marked as intended property was the "butterfly-ballots" used in the 2000 Florida presidential election. Holes in which the voter marked their intent did not align well with their corresponding candidates causing many voters to mark holes that did not correctly

reflect their voting intent. The ambiguity in the layout of this ballot was shown to "have a significant impact on the final election result," [65].

Recorded as Marked

Assuming that the voter correctly understood and executed the marking procedure, the system must make a correct account, or record, of the mark. Here the responsibility rests primarily on the voting equipment manufacturer. It is easy to see that this is of least concern in the paper ballot example; placing a correctly marked paper ballot into the ballot box constitutes correctly recording the mark. However in the case of an electronic system, the human participants of the election cannot physically observe the contents of the electronic memory. A physical analogy of this property can be found when crossing the street. Pushing the button at the base of the traffic pole does not necessarily induce a pedestrian "walk" signal to appear on the next traffic light cycle. Often it does not, causing one to wonder if the button's electrical leads are dangling loose inside the pole. Indeed, could you conclusively prove they were not? Interestingly because of the property of ballot secrecy, it cannot be directly proven that ballots in a live election were correctly recorded as marked–except individually, by the voter.

Counted as Recorded

The final step of the process concerns the aggregation of the recorded votes into what will become the election's outcome. Here the election administration is responsible for the correct counting of votes, but the voting equipment manufacturer also plays a significant role. In traditional hand counted paper ballot elections, a contingent of independent election observers (scrutineers) witness the counting procedure to mitigate the potential for fraud or error. Once again we can see how the use of electronic machines confound this process by their lack of observability. Cases of incorrect tabulations due to error or fraud have been reported in live elections. According to one report, a county in Florida (of a few hundred eligible voters) registered *minus* 16,022 votes cast for Al Gore during the 2000 United

States presidential race, [34]. Clearly counting fraud/errors that results in an impossible outcomes can be detected. But what about counting fraud/errors that produce possible (especially plausible) yet incorrect results?

2.1.2 Privacy in Voting

In the context of the voting problem in a secret ballot election we define privacy in terms of the ability of an unauthorized entity to link votes with identities of voters. We define a dichotomy of privacy centered around the desire of the voter to reveal information about their vote, which we refer to as:

- **Voluntary Privacy**,
- **Involuntary Privacy**.

Traditional privacy enhancing technologies such as anonymizing mix networks [12] carry an assumption that the user wishes their privacy to be protected through the use of the technology. Because the use of the technology is voluntary, the user is at liberty to relinquish their own privacy. The scope of elections is more restrictive. In addition to protecting the voter's privacy from external entities, the election administration is tasked with preventing the voter from voluntarily violating their own privacy. The study of privacy in this context means *technology and procedure that prevents the voluntary or involuntary disclosure of information about a voter's vote.*

Voluntary Privacy

Voluntary privacy refers to the inability of the voter to intentionally disclose information about how they voted. We do not concern ourselves with statements the voter might make about their vote, only that they do not have the ability to prove them. The most prominent reason to prevent voters from voluntarily violating their own privacy is to protect against the undue influence of vote buying. Vote buying can carry several increasingly strict definitions [60]. For our purpose we will define it strictly as an illegal contract between

a vote buyer and a voter. In an election where procedures protect against the voter voluntarily violating their privacy, the voter is left with no proof the contract was fulfilled, meaning the vote buyer can only ever enter the contract in "good faith." Yet in an age of portable imaging devices such as cellphone cameras, this can be difficult to enforce. One practical scenario in which a paper ballot election can become subject to massive vote buying is through a technique known as chain voting [37]. If one voter is able to remove an uncast ballot from the polling area undetected, then the vote buyer is able to witness the desired mark placed on that ballot. This ballot is given to a second voter who enters the polling area and receives a second unmarked ballot. Instead of marking and casting this ballot, the second voter casts the first marked ballot, and removes the unmarked ballot from the polling area, allowing the cycle to continue indefinitely. In practice vigilance on behalf of the poll staff is often sufficient to prevent chain voting. In the case of an E2E election however, further attention must be paid to the design of the ballot receipts–it must not contain information that might be eventually used for the purposes of recovering the vote.

Involuntary Privacy

Involuntary privacy refers to the inability of an entity other than the voter to acquire information about how a voter voted. This is not nearly as great of a concern in a properly conducted paper ballot election, assuming ballots do not contain unique information. In practice however paper ballots will contain some identifying information. Fingerprints of the voter, or an identifying mark made by the voter are two such possibilities. However the election procedure itself can be guilty of introducing identifiable information onto ballots.

Attacks on Electronic Elections Many elections mandate maintaining a poll record– a document which lists the names (and order) of voters as they vote. It is clear in the case of electronic voting machines, being computational devices (and hence deterministic by nature), that the order in which votes are recorded in memory is also deterministic and therefore potentially recoverable. If votes are stored in memory in the order they were

recorded, combining the order of votes in memory with the order of names listed in the poll record would introduce a link between vote and voter. Though this is not inherently the case in all electronic systems, it is also not easy to prove to voters that ordered recording did not occur. Indeed designing a securely random memory map[2] is a non-trivial design feature. Indeed it was recently revealed, among a list of other security vulnerabilities that the widely used Diebold AccuVote-TSX (touchscreen voting machine) "fails to protect ballot secrecy" and stores ballot information in memory in a known ordering [8].

Attacks on Paper Ballot Elections While much attention is given to electronic voting machines, there is often little discussion of how the privacy of paper ballot systems could be violated. As part of a study conducted for the purpose of this thesis in conjunction with the Elections Ontario poll-worker training manual [26], a potential attack avenue for the violation of involuntary privacy was found in the 2007 Ontario provincial election. According to procedure, poll workers are required to tear a ballot off a pad of ballots along a perforated edge. Though the ballot itself does not contain a serial number, the stub on the other end of the perforation does. Additionally a pollbook is kept recording the names (and order) of voting individuals. The ballots, serial stubs and poll record of each poll (of about 300 eligible voters) are individually archived by elections Ontario. With a search space of only 300 pairs, high resolution digital images of perforations could be used to match fibers of the ballot and stub's perforated edges to link marked ballots to their serial number. Assuming the ballots were distributed to voters by ascending serial number, links between serial number and voters name in the poll book can be made. It is proposed by this thesis that anyone with custody of the poll records could, with desktop hardware and software, link ballots to voter names in the 2007 Ontario Election.

[2]Storing votes in an order that would not allow any entity (include the equipment vendor) to determine the order the votes were recorded in.

2.2 Survey of Vote-counting Systems

For our purposes we define a "vote-counting system" as constituting the *enabling process* used to record and aggregate voter intent in an election. This is to be distinguished from the term "voting system" which refers to the *rules* used to mark and aggregate voter intent in an election. In that sense a *vote-counting system* is similar to a hand-calculator whereas a **voting system** is similar to an arithmetic operation. Unfortunately these terms are often used interchangeably in literature.[3]

An illustrating example of a voting system (as defined by this thesis) is the "single-member plurality," or "first past the post" (FPTP) system used in Canadian federal elections. The voter makes a single mark for their most preferred candidate, and the single candidate with the plurality (i.e., majority) of marks is elected. Other common examples of voting systems include, "mixed-member proportional," "condorcet," and "runoff." Refer to [47, 15] for definitions of such systems. Conversely the *vote-counting* system used in Canadian federal elections is a version of the hand-counted paper ballot, also known as the "Australian ballot." Punchscan itself is an optical scan *vote-counting* system and additionally an *independent verification (E2E)* system which supports voting on several *voting systems* (including single and mutliple member pluralities as well as ranked voting).

For the purposes of this document, we take a system-level view of a vote-counting system and have defined it to be consistent with the integrity goals outlined in the previous section. Therefore we define a vote-counting system to consist of a set of three (physical or informational) functional units:

1. Intent capture,
2. Intent storage,
3. Intent aggregation.

Intent capture and storage fall under the "recorded as marked" integrity criterion, while

[3]For example the Brennen Center reports on "voting systems" [5, 6] focus on what we refer to here as "vote-counting" systems.

proper intent aggregation satisfies the "counted as recorded" criterion. Before we address the system design of Punchscan we will survey the most common vote-counting systems.

2.2.1 Australian Ballot

Also known as the "secret ballot," the Australian ballot saw its first use in 1856 in that country. It is a term that is in common use and can be found in many English-language dictionaries. The original usage came from the distinguishing property that the ballot was marked and cast in secrecy. With the advent of mechanical and now electronic vote-counting machines, the term today refers more to the pre-printed paper ballot and manually tabulated nature of the system. The functional units of this system are physical and manual. Intent is indicated on a paper ballot. The voter uses a writing implement such as a pencil or pen to mark their voting intent. Intent is recorded by the system when the voter physically places the ballot into the intent storage unit. Intent storage houses the paper ballots in a physical enclosure such as a cardboard "ballot box." Finally intent aggregation is performed manually by poll workers.

In single-contest elections (such as Canadian federal elections) voting, counting and reporting of results can typically be completed in one day. With proper administration and oversight the relatively low equipment cost and high observability of the process makes this an ideal system for use in developing countries. Though still in wide use around the world today, the Australian ballot system has been in steady decline and almost abandoned in the United States in favor of automated counting systems (primarily those described below). Paper hand counted ballots in the US were used by only 1.6%, 0.6%, and 0.2% of voters in 1998, 2004, and 2006 respectively, [28, 24, 25]. To these districts, the Australian ballot is considered too cumbersome to manage federal elections in which ballots typically contain two dozen simultaneous races for varous public offices.

2.2.2 Direct Recording Mechanical

Lever machines are a mechanical device in which voters mark their choices using mechanical switches. Popularized in the 1950's, to cast their vote the voter flips a mechanical lever which registers the vote in memory by incrementing mechanical counters through a gear system. The mechanical innards of this device resemble that of an automobile odometer. Long criticized for reliability (and security) issues [36], the use of lever machines was banned in New York state as of September 2006 [21]. Punched cards, another mechanical recording system required voters to punch holes in the ballot in perforated regions corresponding to their choice. Similar to the way in which computers of the 1960's and 70's were programmed, the punched cards were fed through an electro-mechanical "card reader" to record the vote. However the usability of Punched cards have been the subject of great concern over intent transcription (i.e., marked as intended) errors. Lever and punched card systems saw a sharp decline of usage in election precincts, falling from 26% to 13% of the registered voting population between the 2004 and 2006 elections [24, 25].

2.2.3 Optical Scan

The functional units of this system are a combination of physical and informational. Intent input is acheived with a paper ballot identical in form of the Australian ballot with the exception of technology enabling formatting (such as scanner alignment marks). The voter still uses a writing implement such as a pencil or pen to mark their voting intent. As its name suggests, the intent recording unit consists of an optical scan device that senses and electronically encodes the physical locations of marks made on the paper ballot. Though an optical scan system typically retains the physical ballot, intent is stored on a computer memory device. The physical record is only referred to in audit/dispute situations. Finally intent aggregation is performed electronically either on the individual device or by physically removing the electronic memory unit and bringing it to a centralized counting facility. Though the recording, storage and aggregation are performed electronically, and therefore unobservably, because optical scan systems usually retain the original physical record, the

integrity of the system typically doesn't fall under as much scrutiny as fully electronic (e.g., touch-screen) systems. Typically lower in cost, optical scan is now the most widely used system in the United States for voting (used by 56% of American counties in 2006, [24]).

2.2.4 Direct Recording Electronic

Direct Recording Electronic (DRE) vote counting systems are a controversial yet popular technology, allowing voters to vote on a computer touchscreen. Despite security concerns and a high purchase price they are regarded as being highly usable allowing voters to confirm or edit their selections before casting their vote. The functional units of this system are primarily informational. Ballots are displayed on a digital display device and intent is recorded through a haptic interface such as a touchscreen. Like the optical scan system, intent is also stored on an electronic memory device and aggregated by a computer system. However because there is no original paper recored to consult, the privacy and integrity of these systems has drawn extensive criticism from researchers, policy and law makers alike, [62, 2, 9, 4, 5]. The EAC has mandated that DREs be used in conjunction with VVPAT [22, 23], though as was stated in the introduction, this is itself an imperfect solution. Despite the growing concern over the use of these machines, DRE systems have seen a steady increase in use, growing almost as quickly in popularity as optical scan systems. Use as a percentage of voting population grew from 29% to 38% across the 2004 and 2006 elections, [24, 25].

2.3 Independent Verification

Much of the concern over electronic voting systems has been a fundamental inability to directly observe or verify the contents or operations on data in electronic memory. Instead of voters trusting equipment vendors, or even government reviews, this thesis proposes a more direct approach. Independent verification (IV) is ability of an individual or organization (unassociated with the election trustees) to undertake a proof of correctness of election results against fraud or error.

The administration of a conventional optical scan based election, which maintains a paper record (of the original ballots), typically does not refer to this record except in the event an irregularity is detected. However irregularities (especially those introduced through fraud) would not be expected to be detectable except by an audit, introducing a paradox into the audit procedure. In the event an audit is undertaken, only a small sample of ballots are selected (presumably at random) for the occasion [18]. Ultimately however these audits are conducted by the election administration and not the public. Yet even if this paper record were made wholly and publicly available, the independent auditors would not be able to establish the authenticity of the ballots they were presented with. Conversely, independent verification through Punchscan is:

- Performed on (a component of) every ballot.
- Performed at every election.
- Available to and repeatable by anyone.
- Open specification, open source
- Software/hardware independent
- Easy enough to perform that it encourages widespread participation

Independent verification is at the heart of the Punchscan system. The results of a Punchscan election are verifiable by voters through their participation in the audit process, a process that establishes a high statistical degree of confidence in the integrity of the outcome.

Prior to the election, the election authority prints the ballots. Each ballot will contain unique information; a serial number and data used for encryption/decryption. This information is "committed to" using a cryptographic bit-commitment function. A discussion of how to construct such functions can be found in [39]. Though these commitments will be publicly posted, the actual information contained on the ballot remains secret. Because

the commitment function is one-way, it is computationally infeasible to determine the information on the sealed ballot given only its publicly posted commitment. In Punchscan, receipts are issued to voters to allow them to ensure their mark was recorded correctly in the election. The ballot receipt represents an "encryption" of the voter's vote, and does not allow the voter to prove their vote to anyone.

These ballot receipts (aka "encrypted ballots") will eventually be run through a decryption operation to yield the (unencrypted) election results. These resultant ballots are shuffled, and not directly linked to their original serial numbers.

Independent verification however comes at a price; ballots must contain unique information. Because Punchscan adheres to the traditional principle of the secret ballot, the system must contain additional privacy enabling and assurance components. Therefore when examining the Punchscan system, it is important to keep in mind that we are attempting to design a voting system with the following two properties:

- **Integrity** of election results – through ballot "receipts" and independent verification (i.e. audit).
- **Privacy** of voters' vote – through strong cryptographic design and distribution of trust.

Intuitively one might expect that integrity of election results and ballot secrecy are mutually exclusive. However as we will see, through the use of cryptography and careful design, these two properties are both simultaneously realizable.

2.4 E2E: End-to-End Cryptographic Independent Verification

In 2005, the American Election Assistance Commission (EAC) released a set of voluntary voting system guidelines [22] that includes a description of what they refer to as "End

to End Cryptographic Independent Verification" (E2E) systems. According to the EAC, typical distinguishing features of an E2E voting system are as follows:

- A paper receipt is issued to the voter that contains information that permits the voter to verify that her choices were recorded correctly. The information does not permit the voter to reveal her selections to a third party.
- The voter has the option to check that her ballot selections were included in the election count, e.g., by checking a web site of values that should match the information on the voter's paper receipt.
- Such a system must provide an assurance not only that her ballot choices were correctly recorded (cast-as-intended), but that those selections were included in the election count (counted-as-cast).

In a surprising move the EAC's 2007 follow-up draft of the VVSG removed all mention of criteria, definitions as well as the terms "end-to-end" or "E2E" themselves, [23]. According to those involved in the VVSG's authorship, the most recent draft was unable to produce "directly testable requirements written for E2E at this point," [64]. The EAC however has recommended an "innovation class" of voting system that would "include mechanisms for encouraging submissions of voting systems such as E2E voting systems," [56]. At the time of this writing, a public comment period has been issued by NIST with regard to this innovation class, but it may be years before the term finds itself back into the VVSG. The move has frustrated researchers who are now no longer in common parlance with the EAC on the subject of E2E systems. For the purposes of this thesis however we will continue to consider criteria set forth by the 2005 draft VVSG, with the task of restoring and expanding the E2E criteria left to future academic work.

2.5 Survey of E2E Systems

We've discussed the difference between voting and vote-counting systems, and now we've introduced the notion of IV and E2E systems. As was mentioned before, Punchscan is both

a vote-counting and E2E system. However there are other IV/E2E systems that have been developed. Several proposals have been made in recent years for independent verification voting systems, and the evolution and refinement of these systems is apparent. Early attempts include Chaum's "visual crypto" based ballot proposed in 2004 [13]. The ballot consists of two overlayed sheets that form a matrix of small squares that are pigmented black or white. The candidates name appears as letters composed of a "noisy" collection of these squares. After voting, the voter peels these layers apart, and in so doing, the pixels that compose the letters of the candidate's name are divided (in a random looking way) between the layers. Both layers now have a perceptually unintelligible appearance, and one of the layers is destroyed so that the layers cannot be recombined to reveal the candidate name. The surviving layer is retained by the voter as a receipt. This was a predecessor to Punchscan but suffered from practical issues, especially manufacturing. For this reason ballots were never actually printed and the system was not implemented. The following are notable examples of other systems in the field of independent verification/E2E systems. Important to note however, these systems are all simultaneously E2E *and* vote-counting systems. As a vote-counting system they are all, like Punchscan, examples of optical scan. However Punchscan is the first E2E system to have been fully implemented (August 2006) and successfully used in a binding election (March 2007), the case study of which is presented in this thesis.

2.5.1 Prêt à Voter

The visual crypto ballot was an interesting proof of concept, but in the following year Chaum, Ryan and Schneider proposed a more "practical" system which has become now known as "Prêt à Voter," (PAV) [11]. The system employs a paper ballot, but unlike the Australian ballot which lists candidates in a fixed order, a Prêt à Voter ballot lists candidates in a pseudo-random order across each ballot. Additionally each ballot contains a unique identifier (serial number) and an encrypted object that will be used to "unshuffle" the candidate list later. The voter makes a mark beside the preferred candidate and the

ballot is split in two along a perforation that bisects the candidate list from the marking area. The candidate list is destroyed, and the remaining marked portion is recorded and returned to the voter as a receipt. The encrypted object, known as an 'onion'[4] is passed through a series of "tellers" or MIX nodes to recover the permutation of candidates' names, which will recover the voter's vote.

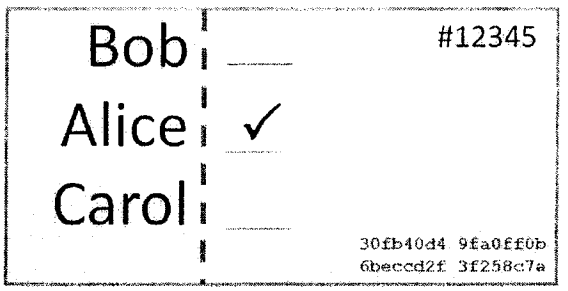

Figure 2.1: **Prêt à Voter ballot**. Left side of perforation demonstrates a randomly ordered candidate list. Right side of perforation demonstrates the marking area (in this case, a vote for Alice) as well as a serial number (top right), and the encrypted onion of the candidate list ordering (bottom right). In practice the onion is far too large to be printed on the ballot (a cryptographic hash of it might appear instead).

The key concept is that the tellers will not be able to recognize the origin of the various layers of the onion as they pass through the nodes, and will not be able to "spy" on the votes. Additionally the tellers themselves can be audited in a way that provides a statistically high assurance of integrity, yet doesn't harm privacy [35]. The system has been refined in subsequent years to provide more robust security and introduce aspects of homomorphic encryption[5] [57, 66], and a working implementation was presented at VoComp 2007. However there is still no concise specification of the audit which would prevent the independent verification criteria from being fully realized until such time as it is made available. Prêt à Voter provides an easy marking experience for the voter, however the practical aspects of MIX net implementation make it more conceptually complex than Punchscan with no security advantage. Additionally some jurisdictions require fixed candidate list orderings which would preclude its use in those locations.

[4]The cryptographic data object of a MIX network, [12].

[5]Homomorphic encryption as it might apply to voting was introduced by Paillier in 1999, [48]. Under this encryption scheme, the (arithmetic) product of encrypted votes, when decrypted, yields their sum.

2.5.2 Scratch and Vote

Scratch and Vote (SnV) was subsequently proposed by Adida and Rivest in 2006, [1]. The SnV ballot is almost identical to Prêt à Voter's except in addition to the encrypted permutation of the candidate list (appearing as barcode), its decryption key is also printed on the ballot. However the key information, being secret, is physically coated in a substance such as that used in "scratch-off" style lottery tickets.

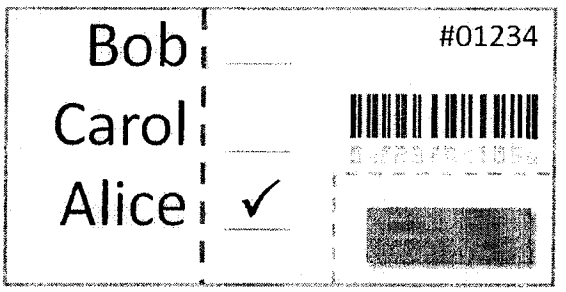

Figure 2.2: **Scratch and Vote ballot**. Left side of perforation demonstrates a randomly ordered candidate list. Right side of perforation demonstrates the marking area (in this case, a vote for Alice) as well as a serial number (top right), a barcode representing the encrypted order of the candidate list (middle right), and its decryption key hidden beneath a scratch-off coating (bottom right). In this example the ballot has been marked, meaning the poll worker would ensure that both the candidate list and the scratch-off portion would be detached and destroyed. An unmarked ballot may be granted to the voter for auditing, in which case the coating is permitted to be scratched off to expose the decryption key. This allows the voter to independently verify that a mark made on that ballot would have been correctly linked to the corresponding candidate.

The voter also cast their ballot in much the same way as PAV except the scratch-off region is detached and destroyed. If the voter wishes to "audit" the printing of a ballot (to ensure the mark will be counted as recorded), the voter can ask for an additional (blank) ballot which will be voided from the election. The voter can scratch off the coating to reveal the key, and use it to decrypt the information displayed in the barcode. Therefore the system would be more accurately described as "Scratch *or* Vote." The system has yet to be implemented.

2.5.3 ThreeBallot

ThreeBallot was proposed by Rivest and Smith in 2007 [55] in conjunction with two other systems (VaV and Twin) as an "educational tool" for IV systems. The intent was a system that was both easy to understand and did not use cryptography.[6]

The ThreeBallot ballot is in fact referred to as a "multi-ballot" as it (as the name suggests) is composed of three conventional (Australian-style) paper ballots joined in a row[7]. Each ballot in the multi-ballot contains a unique identifier (serial number), each of which is not numerically associated with any other serial numbers on any other ballot. To vote, the voter makes a mark for each candidate across the multi-ballot. As there are three ballots, the voter is at liberty to choose from any one of the three ballots to make a mark for a candidate. Finally the voter makes one *additional* mark for their preferred candidate. Explained simply, the Hamming weight across a multi-ballot for every non-preferred candidate is one, while the Hamming weight for the preferred candidate is two. The three ballots are separated from each other, and the voter chooses one of the ballots to be copied (e.g. photocopied) and retained as a receipt. The three ballots are then placed into a ballot box.

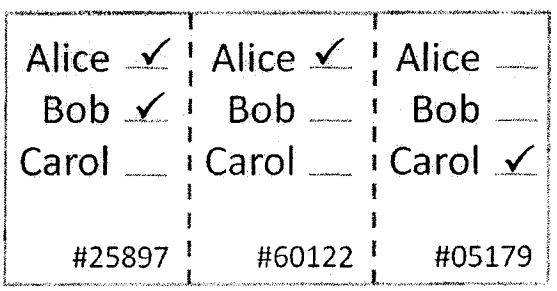

Figure 2.3: **Three Ballot "multi-ballot"**. Three Australian-style ballots joined along perforations each with their own (randomly chosen) serial number. To vote the voter makes a mark for each candidate, and an additional mark for their preferred candidate. Illustrated here is one of several possible marking patterns to indicate a vote for Alice.

To tally the results, the number of marks made for each candidate are summed, and

[6]Though not an E2E system under the definition presented in the previous section, it has been previously referred to by Rivest as "E2E" [54] and is included here for completeness.

[7]Technically ThreeBallot is a voting system as well. The specific procedure for marking a multi-ballot is intended only for single-member plurality (i.e., "first past the post") elections.

the number of multi-ballots cast in the election is subtracted from these sums to yield the election results. To audit the results, the voter looks up their receipt (i.e., *one* of the three ballots from their original multiballot) on an online public bulletin board to verify it was recorded as marked. The most serious (and perhaps most overlooked) drawback of ThreeBallot is the need for ballot validation[8]. In practice validation requires either a DRE (or human being) to prevent overvoting. As a result, the privacy and recorded-as-marked protection is no further ahead than in non E2E systems. Additionally we demonstrated in [16] that there is a potential for bias in ballot marking patterns that would introduce a statistical bias of marks found on a voter's receipt. This introduces a potential (voluntary and involuntary) privacy concern. Furthermore we demonstrated that ThreeBallot does not provide a strong means for securing the bulletin board introducing another avenue for recorded-as-marked fraud.

In this chapter we have introduced the notion of the "voting problem" and defined privacy and integrity criteria that will form the basis of our system design. We highlighted the importance of these criteria by underscoring problems with current vote-counting systems. Then we introduced the notion of independent verification and E2E and gave an overview of three E2E systems.

[8]Without ballot validation a voter could undetectably mark the ballot to cast more than one vote, a practice known as "overvoting."

Chapter 3

Architecture

3.1 Architectural Overview

In this chapter we will begin our examination of Punchscan by presenting the system architecture and its components. To understand the technical and procedural processes presented in subsequent chapters, we must first understand the role each component plays as well as how these components interact.

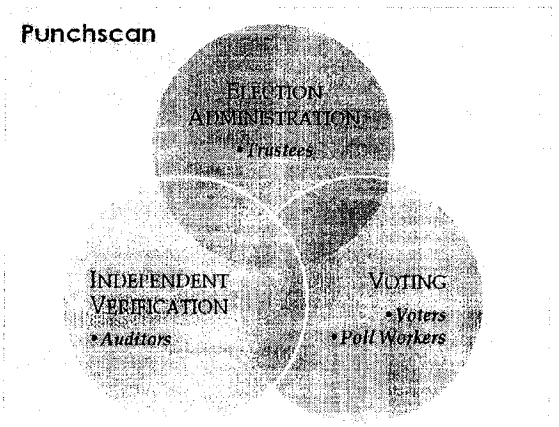

Figure 3.1: System concentrations and user categories.

There are three areas of system concentration. The election administration is managed by the election trustees. Unlike a conventional paper-ballot election, the election administration is responsible for the information security of ballots and ballot receipts as well as

assisting independent verification by producing audit data. The voting-related area is similar to an optical scan election: voters mark their intent on paper ballots and poll workers record their intent with an optical scan device. Independent verification is performed by independent verifiers whom we don't explicitly specify here, but realistically comprise of a subset of voters. The independent verifiers use software tools to audit the election and to check their receipts were recorded correctly.

We see in figure 3.2 an expanded view of the three system concentrations, illustrating the primary system components and their interaction. In the following sections we will discuss the components of these three areas of concentration:

- Voting Components
    1. Punchscan ballot
    2. Polling Place
- Election Administration Components
    1. Punchboard
    2. Diskless Workstation
    3. Trusted printer
- Independent Verification Components
    1. Public Bulletin Board
    2. Pre/post election audit tools
    3. Receipt Checker
    4. Audit Challenge Generator

3.2 Voting Components

In this section we discuss the main components of the voting area of concentration. These include the poll place equipment, but particular emphasis is placed here on Punchscan's

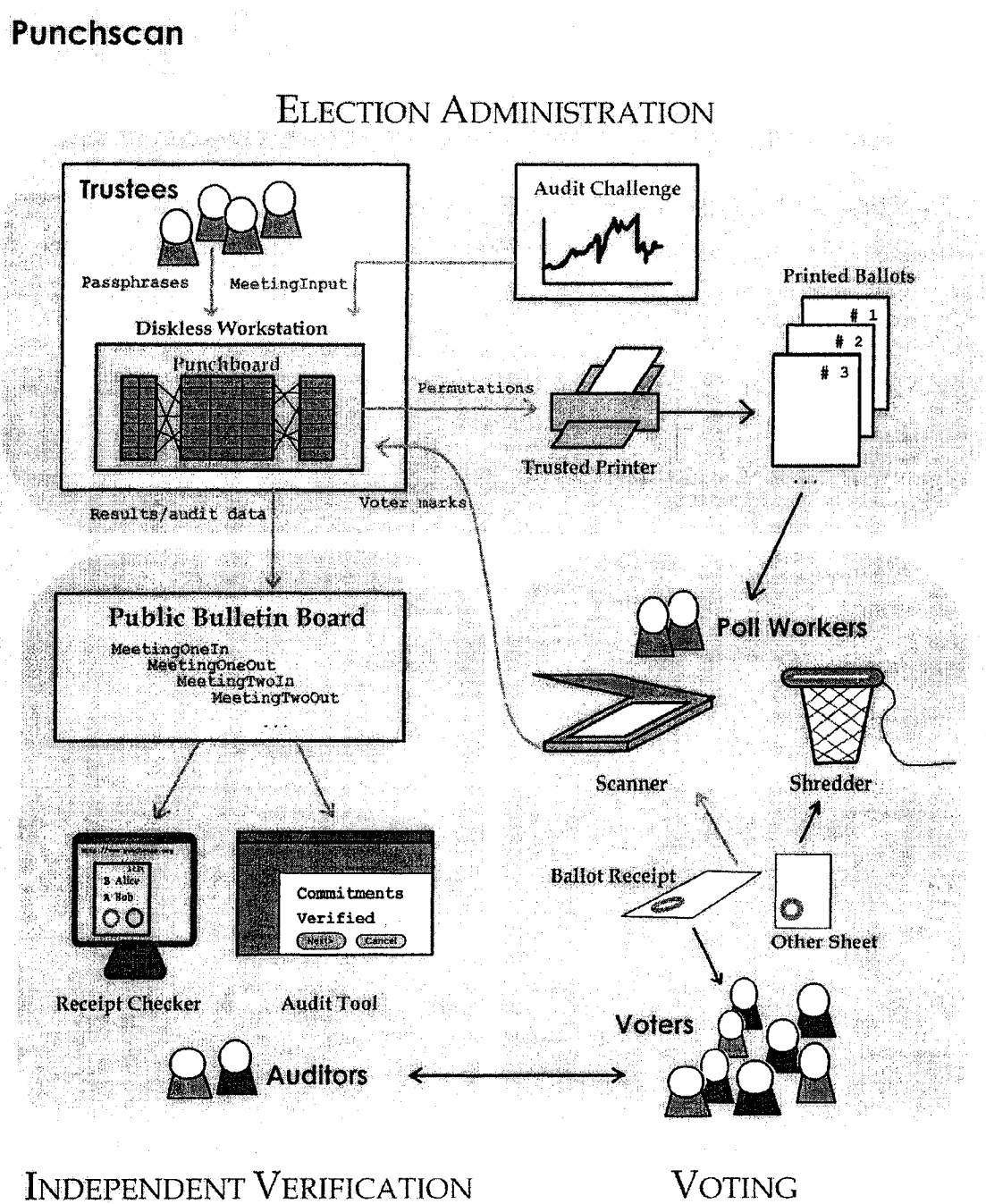

Figure 3.2: System architecture. **Top:** Election administration. **Bottom left:** Independent Verification. **Bottom right:** Voting.

most distinguishing component the ballot.

3.2.1 The Punchscan Ballot

Physical Description

The Punchscan ballot, as illustrated in Figure 3.3 is consists of a top and bottom sheet of paper. The top sheet has a set of symbols next to candidate names as well as holes in it. The bottom sheet also has a set of symbols which are positioned to show through the holes in the top sheet. The letters on both sheets are ordered in a "random looking" (i.e., deterministic pseudo-random) way.

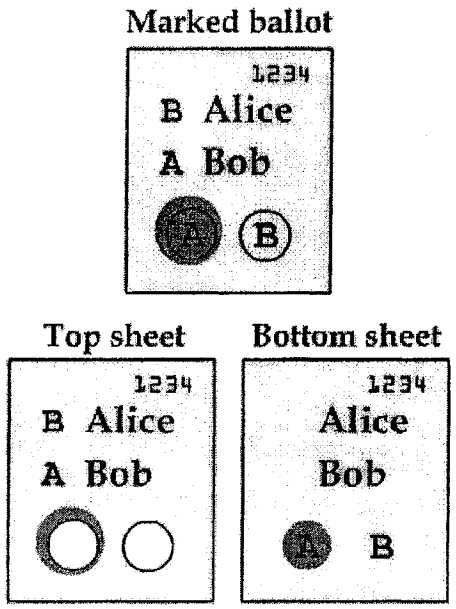

Figure 3.3: A simple Punchscan ballot showing a single two-candidate race. The distinguishing features include a fixed-order candidate list, a unique serial number and a set of symbols written on two overlayed sheets of paper. The symbols on the top sheet are printed next to the candidate list. The symbols on the bottom sheet are printed to show through a set of holes punched in the top sheet. The symbol set used on both sheets is the same, but the ordering is independently random on both sheets. This diagram illustrates a mark that would indicate a vote for "Bob."

To mark a ballot the voter uses an ink dauber (e.g., common "bingo" style dauber) to make a mark centered over one of the holes. The dauber is sized to be slightly larger than the hole such that ink with be deposited on both the top and bottom sheets in the same

position. To make a mark for their chosen candidate the voter must:

1. Note the symbol appearing next to the chose candidate's name,
2. Locate the like symbol appearing in one of the holes,
3. Gently press the ink dauber over that hole.

The two pages are separated, and one of the sheets will be selected to be destroyed (e.g., shred in a cross-cut paper shredder). The remaining sheet is scanned and returned to the voter to be kept as a "receipt." Because the ordering of the symbols on the top and bottom sheets are independent[1], having access to only one sheet will not allow anyone to determine for whom the voter voted. However, the receipt does represent the original (and physical) copy of a voter's mark. This receipt can ultimately be used to verify the recorded as marked integrity criterion (while still preserving the privacy criteria).

Functional Description

Let us define the ballot, its marking and its "decryption" in a more formal manner. In a traditional (Australian, Opscan or DRE) style ballot, voter intent is indicated by the *position* of a mark representing a position (i.e. index) in a canonical list of candidates. For example a mark made in the first position would indicate a vote for the candidate named in the first position on the ballot, etc. Punchscan "encrypts" the voter's intent by mapping the canonical index of candidates to a pseudo-random ordering of positions. Because there are independent orderings on both sheets, seeing a mark on one sheet will not reveal the vote without knowledge of the other sheet.

The ballot Let list $L = \{\omega_0, \ldots, \omega_{n-1}\}$ define a set of n candidates that will appear on the ballot. Let $i_c = \{0, \ldots, n-1\}$ represent the positions (i.e., index) in a list of candidates and let $\kappa : L \rightarrow i_c$ represent a canonical mapping of candidate names to their index. This list is fixed meaning the order of the candidates list printed on each ballot is done with the

[1]Symbol orderings are statistically independent, though cryptographically related (i.e. determined by a secret key).

same κ. Let $i_h = \{0, \ldots, n-1\}$ refer to the index of the n marking regions (i.e., holes) on the ballot. Let us define a symbol alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ (which in practice has been the Latin alphabet, $\{A, B, \cdots, Z\}$). Each individual Punchscan ballot will contain two (pseudo-randomly chosen) bijective mappings. The mapping on the top page will map the canonical index of each candidate's name to a symbol, $P_T : i_c \rightarrow \Sigma$. The mapping on the bottom page will bijectively map symbols to a corresponding marking region, $P_B : \Sigma \rightarrow i_h$.

Marking the ballot Let the candidate name whom the voter wishes to vote for be expressed as $\omega_v \in L$. In order to mark the ballot for ω_v the voter must establish and mark the position $M \in i_h$ where

$$M = (P_B \circ P_T \circ \kappa)(\omega_v).$$

Recovering intent Let decryption permutation $D : i_h \rightarrow i_c = (P_B \circ P_T)^{-1}$. Recovering the voter's intent from the position of their mark i_h is defined as

$$\omega_v = (\kappa^{-1} \circ D)(M).$$

Punchscan uses *two* decryption permutations D_R and D_L where $D = (D_R \circ D_L) = (P_B \circ P_T)^{-1}$ and likewise

$$\omega_v = (\kappa^{-1} \circ D_R \circ D_L)(M).$$

Of note however, although $(D_R \circ D_L \circ P_B \circ P_T)(i_c) = i_c$, D_L is chosen pseudorandomly and independent of P_T, P_B. D_R is then selected to satisfy the above relationship. The reason for this two stage decryption process is developed in the following section.

3.2.2 Polling Place

The polling place is the location where the ballots are distributed to, and marked by, the voters. It is also where the poll workers record (i.e. capture) voter intent.

Paper Shredder As was mentioned for the purposes of privacy (for each ballot cast in the election) one of the ballot sheets is shred. The polling location requires the presence of a high quality (e.g., cross cut style) paper shredder for this purpose.

Optical Scan After one of the ballot sheets is shredded, the other sheet is optically scanned to record the serial number and position of marks made by the voter. Because the sheet is "encrypted," the platform performing, recognizing and storing the optically scanned marks will not be able to discern how a voter voted.

3.3 Election Administration Components

In this section we examine the components used by the election trustees for the purposes of election administration. Particular emphasis is placed on developing a design rationale for the cryptographic data structure, known as the Punchboard, which is used to encrypt and decrypt ballots in a way that offers privacy preserving integrity assurance. Then we briefly discuss how this Punchboard is generated and how the ballots (that reflect aspects of the Punchboard) are printed.

3.3.1 The Punchboard – Specification and Design Rationale

The Punchboard is the core data structure of the Punchscan election and defines how ballots will be printed, voted and decrypted. It has been modularly designed to allow portions of it to be revealed for the purposes of verifying integrity while preserving privacy. The name stems from David Chaum's original analogy of a large publically viewable bulletin board. The ballot encryption and decryption parameters would be written in appropriate cells on the bulletin board and be covered up with tape, "committing" those parameters to use in the election. Later as audit challenges were issued, some of the tape would be removed to expose the data. Previous descriptions of the Punchboard [51] [30] [29] have taken the approach of simply presenting its structure without particular attention to justification of core design decisions. This approach may risk leading the reader to the incorrect conclusion

that portions of this structure are arbitrary or unnecessary. For the sake of clarity, in this treatment of the Punchboard we will build the concept up step by step.

Encrypted Receipt with Verifiable Decryption

Here we present a trivial scheme to demonstrate the incorporation of an encrypted receipt into an election. It offers no privacy. In an election optimized for maximal integrity, voter intent must be coupled with uniquely identifiable information. The example stated earlier was a "show of hands" election where voter intent was directly linked to voter identity. This model does not inherently preclude the issuance of a ballot receipt, but the assumption is that a receipt would be a simple copy of the ballot (e.g., a raised hand). Obviously the show of hands example is incongruent with the secret ballot principle. We can take a step toward privacy by encrypting the vote and issuing a receipt of the encryption. Let E_k represent a bijection where

$$\forall c_i, \exists p_i | E_k(p_i) = c_i,$$

and such that is computationally infeasible for a probabilistic-polynomially time [31] entity, given c_i, to compute a $p_i | E_k(p_i) = c_i$ without knowledge of k. We will refer to this function as an *encryption* function and refer to k as the encryption function's "key". Specifically we will consider the class of "symmetric" encryption functions where an inverse (i.e. decryption) function D exists where $\forall p, D_k(E_k(p)) = p$.

Given a voter's voting intention v, we can define a unique receipt R where $R = E_k(v)$ that the voter could later use to prove or disprove that v was recorded as marked. Consider a bulletin board that lists the receipt values R, the decryption key of R and the reported vote v':

Encrypted Receipts	Decryption Key	Result
$R_1 = E_{k_1}(v_1)$	k_1	$v'_1 = D_{k_1}(R_1)$
$R_2 = E_{k_2}(v_2)$	k_2	$v'_2 = D_{k_2}(R_2)$
$\vdots$	$\vdots$	$\vdots$
$R_n = E_{k_n}(v_n)$	k_n	$v'_n = D_{k_n}(R_n)$

Table 3.1: Bulletin board for encrypted receipt with verifiable decryption

Assuming receipt R_i correctly reflects the encryption of the vote v_i with key k_i the voter can verify that the vote recorded by the election authority, v_i', by decrypting R_i with k_i. Initially the decryption key column would not be public. Later as voters challenged their receipts to be audited, they would request to see their decryption key k_i to verify that

$$D_{k_i}(R_i) = v_i'.$$

It is easy to see there are several problems with this model. Firstly, the receipt does not protect privacy since there is a clearly indicated link between R_i and v_i'. Consider for a moment a scenario where the Receipt and Results column are not directly linked. The voter could not verify the decryption (and hence the encryption) was properly performed unless they knew key k_i. However knowing this key *re-introduces* the link between receipt and result, violating our privacy criteria. Essentially we are attempting to verify a chain without being allowed to know, at the same time, where it begins and where it ends.

Encrypted Receipt with Privacy Preserving Verifiable Decryption

Here we present an improved scheme that demonstrates how to verify correct decryption of an encrypted receipt in a way that preserves privacy. Although this feature itself offers privacy across the verification of a ballot's decryption, the system as a whole still does not. We require an encryption scheme that can be verified for correctness but does not require the *simultaneous knowledge* of both the ciphertext and plaintext (in this case R_i and v_i'). The first improvement that can be made is to split the decryption process into two steps. This is the fundamental yet powerful feature of the Punchboard. Consider a bulletin board in which a ballot is originally encrypted under two keys k_a and k_b, and then decrypted in two steps with a column for the intermediate result I:

For each row in the table, R, I and v' are publically known whereas all instances of k_a and k_b are initially not public (i.e. secret). To verify the correctness of the decryption (and encryption), for each row in the decryption table either the "left" or "right" side can be challenged for audit. That is to say, the audit challenges compels the election authority

	Decryption Table			
Encrypted Receipts	Key k_a	Intermediate Result I_n	Key k_b	Result
$R_1 = E_{k_{a_1}}(E_{k_{b_1}}(v_1))$	k_{a_1}	$I_1 = E_{k_{b_1}}(v_1) = D_{k_{a_1}}(R_1)$	k_{b_1}	$v'_1 = D_{k_{b_1}}(I_1)$
$R_2 = E_{k_{a_2}}(E_{k_{b_2}}(v_2))$	k_{a_2}	$I_2 = E_{k_{b_2}}(v_2) = D_{k_{a_2}}(R_2)$	k_{b_2}	$v'_2 = D_{k_{b_2}}(I_2)$
		$\vdots$		
$R_n = E_{k_{a_n}}(E_{k_{b_n}}(v_n))$	k_{a_n}	$I_n = E_{k_{b_n}}(v_n) = D_{k_{a_n}}(R_n)$	k_{b_n}	$v'_n = D_{k_{b_n}}(I_n)$

Table 3.2: Bulletin board for encrypted receipt with privacy preserving verifiable decryption.

to produce, for each row i in the decryption table, either k_{a_i} or k_{b_i} with which the verifier can verify either,

$$k_{a_i} : D_{k_{a_i}}(R_i) \stackrel{?}{=} I_i,$$

$$k_{b_i} : D_{k_{b_i}}(I_i) \stackrel{?}{=} v'_i.$$

As long as these challenges are made in a statistically random and unguessable way, any error or modification will be detected with a 50% chance. This forms the other core concept of the Punchboard: "cut and choose" [14]. Although this does not guarantee detection of fraud or error, the probability of detecting a single instance of n instances of fraud/error grows by $P = 1 - 2^{-n}$. The implication of this is that a successful audit (i.e., an audit that did not detect error of fraud) proves with statistical certainty that no (election outcome altering) fraud occured. Clearly however there still exists a link between receipt and vote. This however sets the stage for the next improvement.

Privacy Preserving Encrypted Receipt and Verifiable Decryption

Here we present a scheme that demonstrates how to verify the decryption of a receipt that preserves privacy across the entire system. So far we've introduced a means to verify a link in a chain without needing to know both the endpoints of that chain. The procedure relies on the fact that there are many chains, and that by only proving one half of each chain, we can (with statistical confidence) verify the entire set. However the scheme as it was presented still offers a clear link between the endpoints; the bulletin board lists a receipt

R and the corresponding vote v' in the same row. It is at this point that we introduce the notion of *row-permutations*. Recalling the format of table 3.2, we can "unlink" the receipt, decryption and results columns and permute their order in a (pseudo) random way relative to each other. These permutations will initially be secret, but similar to the technique of revealing only one decryption key for each row, only one permutation will be revealed. In this way the independent verifiers can still verify (given the key and permutation) that the ciphertext correctly decrypts to the plaintext, but without knowledge of the other permutation, cannot determine the terminus of the other half of the chain. Let us define two bijective (i.e. invertable) pseudo-random permutations; a *backwards permutation* P_β and *forwards permutation* P_φ. Integrating these permutations into the decryption table yields the following:

	Decryption Table					
Encrypted Receipts	Backward Row Permutation	Key k_a	Intermediate Result	Key k_b	Forward Row Permutation	Results
$R_1 = E_{k_{a_1}}(E_{k_{b_1}}(v_1))$	$\sigma_1 = P_\beta(1)$	$k_{a_{\sigma_1}}$	$I_{\sigma_1} = E_{k_{b_{\sigma_1}}}(v_{\sigma_1}) = D_{k_{a_{\sigma_1}}}(R_{\sigma_1})$	$k_{b_{\sigma_1}}$	$\varsigma_1 = P_\varphi(1)$	$v'_{P_\varphi^{-1}(1)} = D_{k_{b_{P_\varphi^{-1}(1)}}}(I_{P_\varphi^{-1}(1)})$
$R_2 = E_{k_{a_2}}(E_{k_{b_2}}(v_2))$	$\sigma_2 = P_\beta(2)$	$k_{a_{\sigma_2}}$	$I_{\sigma_2} = E_{k_{b_{\sigma_2}}}(v_{\sigma_2}) = D_{k_{a_{\sigma_2}}}(R_{\sigma_2})$	$k_{b_{\sigma_2}}$	$\varsigma_2 = P_\varphi(2)$	$v'_{P_\varphi^{-1}(2)} = D_{k_{b_{P_\varphi^{-1}(2)}}}(I_{P_\varphi^{-1}(2)})$
$\vdots$						
$R_n = E_{k_{a_n}}(E_{k_{b_n}}(v_n))$	$\sigma_n = P_\beta(n)$	$k_{a_{\sigma_n}}$	$I_{\sigma_n} = E_{k_{b_{\sigma_n}}}(v_{\sigma_n}) = D_{k_{a_{\sigma_n}}}(R_{\sigma_n})$	$k_{b_{\sigma_n}}$	$\varsigma_n = P_\varphi(n)$	$v'_{P_\varphi^{-1}(n)} = D_{k_{b_{P_\varphi^{-1}(n)}}}(I_{P_\varphi^{-1}(n)})$

Table 3.3: Bulletin board for privacy preserving encrypted receipt and verifiable decryption

For each row in the table, R, I and v' are publically known whereas all instances of σ, k_a, ς and k_b are initially not public (i.e. secret). To verify the correctness of the decryptions and permutations, for each row in the decryption table either the "left" or "right" side can be challenged to be audited. That is to say, the audit challenges compels election authority, for each row i in the decryption table, to produce either $\{k_{a_i}, P_\beta(i)\}$ or $\{k_{b_i}, P_\varphi(i)\}$ with which the verifier verifies either,

$$\{k_{a_i}, P_\beta(i)\} : D_{k_{a_i}}(R_{P_\beta^{-1}(i)}) \stackrel{?}{=} I_i,$$

or

$$\{k_{b_i}, P_\varphi(i)\} : D_{k_{b_i}}(I_i) \stackrel{?}{=} v'_{\varsigma_i}.$$

In summary this verification process allows independent verifiers to globally establish counted-as-recorded accuracy without linking receipts to votes, satisfying the voluntary/involuntary privacy criteria. The integrity criteria are reasonably satisfied across the aggregate because in that election outcome altering errors will be detected (with high statistical certainty. But what about integrity across the individual ballot? Though aggregate integrity is strong, it is reasonable to conclude that a 50% probability of catching an error in the decryption table of an individual ballot remains unsatisfactory in terms of integrity assurance. Next we propose a scheme to enhance integrity assurance across individual ballots.

Multiple Instances of the Decryption Table

Recall that the decryption table was modularly designed, meaning that either of the "left" or "right" sides could be published for audit purposes without revealing how the voter voted. This means however that if an error or intentional fault were introduced into either one of these sides, it would only be detected with a 50% chance. Obviously however it would be peferable that the probability of detection was overwhelming. Again Punchscan provides a solution in an exceedingly simple yet powerful way: generate multiple indepedent instances of the decryption table.

Recall from the discussion of the Punchscan ballot that each ballot i has top and bottom random permutation P_{T_i} and P_{B_i} respectively. To recover the vote ω_{v_i} from mark M_i these decryptions are applied as,

$$\omega_{v_i} = (\kappa^{-1} \circ D_{R_i} \circ D_{L_i})(M_i),$$

where functions D_{R_i} and D_{L_i} are related to P_{T_i} and P_{B_i} by $(D_{R_i} \circ D_{L_i}) = (P_{B_i} \circ P_{T_i})^{-1}$. Consider as scenario in which n decryption pairs were generated;

$$\begin{aligned}(P_{B_i} \circ P_{T_i})^{-1} &= (D_{R_i}^{(1)} \circ D_{L_i}^{(1)}) \\ &= (D_{R_i}^{(2)} \circ D_{L_i}^{(2)}) \\ &\vdots \\ &= (D_{R_i}^{(n)} \circ D_{L_i}^{(n)}).\end{aligned}$$

There would be correspondingly n versions of intermediate result

$$\left\{I_i^{(1)}, I_i^{(2)}, \cdots, I_i^{(n)}\right\},$$

as well as n versions of row permutations,

$$\left\{P_\beta^{(1)}(i), P_\beta^{(2)}(i), \cdots, P_\beta^{(n)}(i)\right\}$$

and

$$\left\{P_\varphi^{(1)}(i), P_\varphi^{(2)}(i), \cdots, P_\varphi^{(n)}(i)\right\}.$$

This data would be arranged as n independent instances of the decryption table. The results table would remain unchanged, meaning of course one version of v_i'. To audit a ballot, instead of one challenge $\{D_{L_i}, P_\beta(i)\}$ or $\{D_{R_i}, P_\varphi(i)\}$ being issued, n challenges (one for each instance of the decryption table) are issued:

$$
\begin{aligned}
c_1 &= \left\{D_{L_i}^{(1)}, P_{\beta(i)}\right\}, or, \left\{D_{R_i}^{(1)}, P_{\varphi}(i)\right\} \\
c_2 &= \left\{D_{L_i}^{(2)}, P_{\beta(i)}\right\}, or, \left\{D_{R_i}^{(2)}, P_{\varphi}(i)\right\} \\
&\vdots \\
c_n &= \left\{D_{L_i}^{(n)}, P_{\beta(i)}\right\}, or, \left\{D_{R_i}^{(n)}, P_{\varphi}(i)\right\}
\end{aligned}
$$

Each of these challenges c_j are performed (separately) to verify either:

$$\left\{D_{L_i}^{(j)}, P_{\beta}^{(j)}(i)\right\} : D_{L_i}^{(j)}(M_{(P_{\beta}^{(j)})^{-1}(i)}) \stackrel{?}{=} I_i^{(j)},$$

or

$$\left\{D_{R_i}^{(j)}, P_{\varphi}^{(j)}(i)\right\} : D_{R_i}^{(j)}(I_i^{(j)}) \stackrel{?}{=} v'_{\varsigma_i}.$$

If v'_i was modified to be anything other than what it should be, it would require altering at least one of $D_{L_i}^{(j)}$ or $D_{R_i}^{(j)}$ for each instance j of the n instances. Assuming once again that the challenges are issued randomly, there is only a 2^{-n} chance that all altered decryptions would escape detection. Therefore the multiple instance of decryption technique offers exponentially increasing integrity assurance at the cost of linearly increasing space/computation. The number of instances of decryption tables is selected as an election parameter and depends on the computing resources available. The GSAED election (presented in chapter 6) used 10 instances of the decryption table on 4000 ballots, and the software audit tool was able to perform all of the cryptographic checks on the order of 2 minutes. To summarize to this point, we have developed an encrypted receipt system with privacy preserving high-integrity assurance across an aggregate of ballots and now across individual ballots as well.

However the discussion thus far has omitted a fatal security gap in this process. The focus has been on verifying the correctness of data. The next aspect of study will be a higher order verification: verifying that the data used for verification was the same data the produced the election results. For us to verify the decryptions, our assumption has

been that posted key/permutation data (as requested in the audit challenge) cannot be changed through the course of the election. As will be explained, this is not inherently the case.

The Role of Cryptographic Commitments

The final, and arguably the most crucial integrity aspect of the Punchboard is its use of cryptographic bit-commitments. Before we discuss what commitments are, we examine why they are necessary. The study of cryptography offers us the notion of *unicity distance*, used to describe the amount of ciphertext that would, given the opportunity to try all decryption keys, yield precisely one valid plaintext. Under normal privacy applications, a ciphertext message greater than the unicity distance of the system is cause for (at least at a theoretical level) concern. With sufficient time and resources an adversary could, with certainty, recover the message. In the scope of integrity verification, precisely the *opposite* is desired. A ciphertext below the unicity distance means there exists more than one key (aka *spurious* keys) that could decrypt the ciphertext to a "valid-looking" plain text. As described earlier, Punchscan uses pseudo-random permutations (i.e. bijections) to encrypt a voter's mark. This means that every key (and resulting pseudo-random permutation) would produce a "valid looking" plaintext. However the creation of these permutations involves a key, and because the verifiers do not (initially) know the key, all keys are spurious from the perspective of the verifier (i.e. the decryption could be manipulated such that any receipt could produce any outcome in the result table).

In considering the verification process of the scheme presented in table 3.3, it is easy to see the election trustees can manufacture any outcome of their choosing, and do so in a way that would (falsely) appear to be verifiable. To execute this attack, the election trustees would simply pre-determine the results R and intermediate results I. At the appropriate time they would post this data on the bulletin board. In response to audit challenges, they would simply select the decryption keys such that any mark made by the voter decrypts to their pre-determined outcome. This key tampering process is trivial and

could be performed by hand.

There is an additional attack vector. Because you're only verifying half of the decryption chain of each ballot, there's nothing to prevent maliciously printed ballots that would register a vote for someone other than for whom the voter marked. We address this issue by randomly inspecting (i.e. spoiling) some of the ballots and their decryption/permutation chain to ensure proper formation (i.e. a mark for a candidate translates to a vote for precisely that candidate). However, nothing prevents the remaining (sealed) ballot from being maliciously modified subsequent to this pre-election audit.

To combat the vulnerability of spurious keys we need to ensure that the ciphertext of the receipt is well above the unicity distance of the system, that is to say, unique. To prevent the pre-election audit attack, we will also need a way to publically "commit" to the permutation and decryption keys before the election, without actually revealing their values. Therefore the injection should be, at minimum, cryptographically secure.

Cryptographic bit-commitments are a means to prove to someone that a secret hasn't been altered through the course of time, while still allowing the secret holder to control the timing of the release of the secret. A physical analogy would be writing a secret on a sheet of paper and placing it in a locked briefcase. The briefcase is placed in the trust of a person you are trying to make the proof to. Initially the individual cannot open the case. At the appropriate time, you hand this individual the key to the briefcase allowing them to open it. In the context of a verifiable election let k represent a bulletin board parameter to be committed to. To address the spurious keys, we introduce a relatively large, uniformly chosen value s, (commonly known as a "salt"). Commitment function C is said to be injective if,

$$\forall \{k,s\}\,\{k^*,s^*\}, C(k,s) = C(k^*,s^*) \Rightarrow \{k,s\} = \{k^*,s^*\}$$

or equivalently if,

$$\forall \{k,s\}, \{k^*,s^*\}, \{k,s\} \neq \{k^*,s^*\} \Rightarrow C(k,s) \neq C(k^*,s^*)$$

This property represents information-theoretic integrity. Additionally to preserve privacy the function C should be (at minimum) computationally one-way, meaning that given $c' = C(k, s)$ it is computationally infeasible to determine the value k without knowledge of s. When the Punchboard parameters are initially generated at the start of an election, commitments of all secret parameters are computed and posted. Later when the audit challenges are issued, the trustees respond by publishing those parameters k along with their salt s. The independent verifiers check this information against the originally posted commitments. The commitment verification protocol is summarized as follows:

Trustees		Indepdendent Verifiers
$c = C(k, s)$	$\xrightarrow{c}$	
	$\vdots$	
	$\xrightarrow{k^*, s^*}$	
		$C(k^*, s^*) = c \iff \{k^* = k, s^* = s\}$

Table 3.4: Commitment verification protocol.

The Punchscan implementation of the commitment scheme (presented in the next chapter) is believed to offer information theoretically assured integrity, and offer computationally assured privacy equivalent to the AES block cipher.

Punchboard Specification

All of the text in this section to this point has been a background and design justification. The encrypted receipt model that we've developed up to this point captures the main features of the Punchboard; two stage decryption for verifiability and unlinkability of results to serial numbers, multiple instances of the decryption table for enhanced verification, and cryptographic commitments for integrity of the Punchboard through time.

Now that the basic framework is in place we turn our attention to the front-end in an effort to address a practical but important issue: how exactly do voters 'encrypt' their votes? The voter could simply be issued a printed receipt of the AES-128 encryption of their vote as was implied by *privacy preserving encrypted receipt and verifiable decryption*

model. Obviously this requires an encryption operation being performed at the polling place introducing serious key-management and software trust issues. Additionally it does not offer strong recorded as marked verification as it would not allow the voter to retain the original mark they made. The Punchscan ballot solves both these problems. Separating and shredding one of the ballot sheets "encrypts" the vote without a computer. Since the remaining sheet is "encrypted" the voter can safely retain it, and thereby keep an original copy of their mark.

For a Punchboard of n ballots and j instances of the decryption table, we define three tables which we refer to as the "print", "decrypt" and "results" tables, or simply "$\{P, D, R\}$-table" for short. The print table contains the ballot "mark" permutations (i.e., the orders the letters are *printed* in). It will also contain the mark made by the voter, and the commitments to the permutations. The decryption table contains the two-stage mark decryption permutations as well as the random "row" permutations. It will also contain the intermediate (partially decrypted) result, and the commitments to the mark decryption/row permutations. Finally the results table will contain the "decrypted" votes, shuffled with respect to the Print table (as specified by the row permutations. The $\{P, D, R\}$-tables are specified in tables 3.5, 3.6, 3.7 respectively.

Print Table					
Serial Number	Top Permutation	Bottom Permutation	Voter's Mark Position	Commitment to P_T	Commitment to P_B
1	P_{T_1}	P_{B_1}	$M_1 = P_{B_1}(P_{T_1}(\kappa(\omega_{v_1})))$	$C(P_{T_1})$	$C(P_{B_1})$
2	P_{T_2}	P_{B_2}	$M_2 = P_{B_2}(P_{T_2}(\kappa(\omega_{v_2})))$	$C(P_{T_2})$	$C(P_{B_2})$
			$\vdots$		
n	P_{T_n}	P_{B_n}	$M_n = P_{B_n}(P_{T_n}(\kappa(\omega_{v_n})))$	$C(P_{T_n})$	$C(P_{B_n})$

Table 3.5: Punchboard Print Table

Initially all parameters with the exception of the commitments are secret. During the pre-election audit, half of the n rows of the P-table (and their corresponding rows of the j instances of the D-table) are opened and verified that $P_B \circ P_T = D_R \circ D_L$. The commitments of each parameter are also verified. Later when polling has completed, the

Decryption Table						
Backward Row Permutation	Left Decryption Permutation	Intermediate Result	Right Decryption Permutation	Forward Row Permutation	Commitment to left side	Commitment to right side
Instance 1						
$\sigma_1^{(1)} = P_\beta^{(1)}(1)$	$D_{L_{\sigma_1}}^{(1)}$	$I_{\sigma_1}^{(1)} = D_{L_{\sigma_1}}^{(1)}(M_{\sigma_1})$	$D_{R_{\sigma_1}}^{(1)}$	$\varsigma_1^{(1)} = P_\varphi^{(1)}(1)$	$C(\sigma_1^{(1)} \| D_{L_{\sigma_1}}^{(1)})$	$C(\varsigma_1^{(1)} \| D_{R_{\sigma_1}}^{(1)})$
$\sigma_2^{(1)} = P_\beta^{(1)}(2)$	$D_{L_{\sigma_2}}^{(1)}$	$I_{\sigma_2}^{(1)} = D_{L_{\sigma_2}}^{(1)}(M_{\sigma_2})$	$D_{R_{\sigma_2}}^{(1)}$	$\varsigma_2^{(1)} = P_\varphi^{(1)}(2)$	$C(\sigma_2^{(1)} \| D_{L_{\sigma_2}}^{(1)})$	$C(\varsigma_2^{(1)} \| D_{R_{\sigma_2}}^{(1)})$
⋮						
$\sigma_n^{(1)} = P_\beta^{(1)}(n)$	$D_{L_{\sigma_n}}^{(1)}$	$I_{\sigma_n}^{(1)} = D_{L_{\sigma_n}}^{(1)}(M_{\sigma_n})$	$D_{R_{\sigma_n}}^{(1)}$	$\varsigma_n^{(1)} = P_\varphi^{(1)}(n)$	$C(\sigma_n^{(1)} \| D_{L_{\sigma_n}}^{(1)})$	$C(\varsigma_n^{(1)} \| D_{R_{\sigma_n}}^{(1)})$
⋮						
Instance j						
$\sigma_1^{(j)} = P_\beta^{(j)}(1)$	$D_{L_{\sigma_1}}^{(j)}$	$I_{\sigma_1}^{(j)} = D_{L_{\sigma_1}}^{(j)}(M_{\sigma_1})$	$D_{R_{\sigma_1}}^{(j)}$	$\varsigma_1^{(j)} = P_\varphi^{(j)}(1)$	$C(\sigma_1^{(j)} \| D_{L_{\sigma_1}}^{(j)})$	$C(\varsigma_1^{(j1)} \| D_{R_{\sigma_1}}^{(j)})$
$\sigma_2^{(j)} = P_\beta^{(j)}(2)$	$D_{L_{\sigma_2}}^{(j)}$	$I_{\sigma_2}^{(j)} = D_{L_{\sigma_2}}^{(j)}(M_{\sigma_2})$	$D_{R_{\sigma_2}}^{(j)}$	$\varsigma_2^{(j)} = P_\varphi^{(j)}(2)$	$C(\sigma_2^{(j)} \| D_{L_{\sigma_2}}^{(j)})$	$C(\varsigma_2^{(j)} \| D_{R_{\sigma_2}}^{(j)})$
⋮						
$\sigma_n^{(j)} = P_\beta^{(j)}(n)$	$D_{L_{\sigma_n}}^{(j)}$	$I_{\sigma_n}^{(j)} = D_{L_{\sigma_n}}^{(j)}(M_{\sigma_n})$	$D_{R_{\sigma_n}}^{(j)}$	$\varsigma_n^{(j)} = P_\varphi^{(j)}(n)$	$C(\sigma_n^{(j)} \| D_{L_{\sigma_n}}^{(j)})$	$C(\varsigma_n^{(j)} \| D_{R_{\sigma_n}}^{(j)})$

Table 3.6: Punchboard Decryption Table

Results Table
$v'_1 = D^{(1)}_{R_{(P_\varphi^{(1)})^{-1}(1)}}(I^{(1)}_{(P_\varphi^{(1)})^{-1}(1)})$
$= D^{(2)}_{R_{(P_\varphi^{(2)})^{-1}(1)}}(I^{(2)}_{(P_\varphi^{(2)})^{-1}(1)})$
⋮
$= D^{(j)}_{R_{(P_\varphi^{(j)})^{-1}(1)}}(I^{(j)}_{(P_\varphi^{(j)})^{-1}(1)})$
$v'_2 = D^{(1)}_{R_{(P_\varphi^{(1)})^{-1}(2)}}(I^{(1)}_{(P_\varphi^{(1)})^{-1}(2)})$
$= D^{(2)}_{R_{(P_\varphi^{(2)})^{-1}(2)}}(I^{(2)}_{(P_\varphi^{(2)})^{-1}(2)})$
⋮
$= D^{(j)}_{R_{(P_\varphi^{(j)})^{-1}(2)}}(I^{(j)}_{(P_\varphi^{(j)})^{-1}(2)})$
⋮
$v'_n = D^{(1)}_{R_{(P_\varphi^{(1)})^{-1}(n)}}(I^{(1)}_{(P_\varphi^{(1)})^{-1}(n)})$
$= D^{(2)}_{R_{(P_\varphi^{(2)})^{-1}(n)}}(I^{(2)}_{(P_\varphi^{(2)})^{-1}(n)})$
⋮
$= D^{(j)}_{R_{(P_\varphi^{(j)})^{-1}(n)}}(I^{(j)}_{(P_\varphi^{(j)})^{-1}(n)})$

Table 3.7: Punchboard Results Table

marks M, intermediate results I and results v' are posted. During the post-election audit, either the left or right side of each row of each instance of the D-table is opened and audited (with these values also being checked against their commitments). In this section we've developed a high-level description of the Punchboard. In chapter 4 we will define the audit data with respect to our implementation and present the algorithms used to verify them.

3.3.2 Diskless Workstation

As was mentioned before, the encryption/decryption permutations are originally secret. Assuming the scheme described above is followed, integrity is assured. If the scheme is not followed in any way, the inconsistancy will be detected with (statistical) certainty during the audit. However for voter privacy purposes it is important that this data is also secret from the trustees. Additionally for privacy it is important that the permutations are created in a statistically unbiased way, meaning that any software producing the Punchboard must posess two properties:

1. Oblivious generation of the Punchboard

2. Verifiable software correctness.

This produces a technical challenge: how can we generate the Punchboard without any trustees' knowledge? What is required is a cryptographic secret-sharing scheme in which a piece of secret information (e.g., a passphrase) known only to each respective trustee can be combined in a cryptographically-secure way to form an election-wide secret key. The key will be input into another cryptographic function to generate the Punchboard parameters. At the end of each operation, only the necessary data is output, and the remaining state of the Punchboard is eliminated (i.e., deleted) from memory. Such a scheme would address the first problem. But who provides the software implementation? Even if the software is open-source (and therefore verifiable, testable and certifiable), how do the trustees guarantee that that precise (i.e., unmodified) version of software is executing? We've chose to address this issue by allowing each trustee to bring a copy of the Punchboard generation software to be

loaded on a trusted hardware platform (a computer with no hard drive). We've developed a software verification scheme the trustees can undertake to cross-verify each other's version of the election software to prove that only one version–the correct version will be executed. The implementation of this solution will be discussed further in the next chapter.

3.3.3 Trusted Printer

The Punchboard as previously described offers strong unconditional integrity of election results. Simply put, those with access to the Punchboard cannot alter the outcome of an election in a way that would be undetectable to the independent verifiers. In contrast, privacy assurance remains an area of development, specifically the involuntary privacy criterion. This centers around the fact that paper ballots contain sufficient information to recover voters' votes. Those with physical access to the ballots are in a position to exploit privacy by recording the top/bottom symbol permutations and serial number. After election day they can use the serial number and voter mark positions to recover a voter's vote. To make this more robust against attack it would be desirable to limit access to ballots. However the entity that prints the ballots has full access to ballot information. In the earlier versions of Punchscan, we've approached this problem by defining a "trusted printing authority" that would print the ballots, but could be relied upon to **not** record the information contained on them.

Unfortunately, this privacy "deus ex machina" is similar to the Three-ballot DRE; if a system with these privacy/integrity properties existed in practice, we wouldn't need to be developing E2E systems to protect privacy and integrity. The easiest way to mitigate the privacy risk of a trusted printing authority is to split its role in half. One facility would be responsible for printing *top* pages, while another unassociated facility would print *bottom* pages. These ballot tops and bottoms would be printed separately and then collated by serial number at the polling station. This obviously still provides the opportunity to the poll workers themselves to see the assembled ballots. A modification was presented in [10] that would allow the top and bottom sheets to be independent of each other, allowing the

voter to randomly select a top and bottom sheet from a respective stack. This would make it difficult for malicious poll workers to surveil individual ballots without knowledge of all ballots. The security assumption is that it would be far more difficult for a poll worker to undetectably gain access to *all* ballots than simply a few opportunistic disclosures. This notion of independent ballot sheets is still an area of development.

The ultimate role of Punchscan however is not to improve voter privacy above current standards, but rather attempt to maintain a comparable standard with existing systems. As we've seen in this section with the discussion of the Punchboard and diskless workstation, the coup de grâce of Punchscan is its integrity assurance.

3.4 Independent Verification Components

Recalling the integrity criteria from chapter 2, we seek to provide voters with recorded-as-marked and counted-as-recorded assurance[2]. The recorded-as-marked criterion will be verified by voters through a receipt check audit that confirms the marks recorded by the election administration match the original marks indicated on voter receipts. The counted-as-recorded criterion is verified by voters via a pre- and post-election audit. During these audits, challenges are made to the election trustees to publically reveal certain pieces of information. This data was previously asserted to constitute some mathematical (i.e. cryptographic) relationship. Open-source software tools are used by voters to verify the existence of this relationship, and therefore the correctness of the data. The following paragraphs merely summarize the roles of pre/post election audits, receipt checker and audit challenge generator, which were initially described in previous sections.

3.4.1 Pre-Election Audit

The pre-election audit ensures proper construction of the Punchboard, and the information that will be printed on the ballots. It is the first of two steps to ensure votes will be counted as recorded. In this step we seek to verify that the choices being presented to the voter,

[2]We presume the marked-as-intended criterion was verified by the voter at the poll both

once encrypted, will be decrypted to *same* choices as they were presented to the voter. During the pre-election audit, *half* the ballots generated by the trustees are selected to be examined. These ballots are published (and spoiled) and checked to ensure they are properly formed (i.e., they will correctly perform the decryption) and that they match their commitments. Given a fair (i.e. randomly chosen) selection, we can be satisfied that the remaining sealed ballots are properly formed.

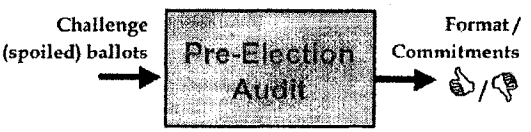

Figure 3.4: Pre-Election Audit

3.4.2 Post-Election Audit

The post-election audit is the second of two steps to ensure votes were counted as recorded. In this step we seek to verify that the marks made by the voter really were decrypted to reflect the choices made by the voter. For each ballot cast in the election, one half of its decryption transformation (either left or right) is unsealed, published, and checked to ensure it correctly performs its half of the decryption. The left partial-decryption step corresponds to the decryption of the ballot receipt to the intermediate result. Conversely the right partial-decryption step corresponds to decryption of the partially decrypted result to the result (i.e. vote). Again, given a fair (i.e. randomly chosen) selection, we can be satisfied that the remaining sealed half of the decryption is valid.

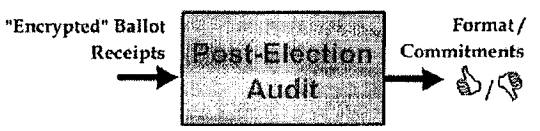

Figure 3.5: Post-Election Audit

3.4.3 Receipt Checker

The receipt check is the process by which the voter verifies that the information contained on their paper ballot receipt matches the information that was used in decryption/tally. This is primarily used to ensure the recorded-as-marked integrity criterion. The receipt check needs to be easy and available to voters to promote their participation in the process. It is however not mandatory for the voter to perform this step, and a high degree of confidence in the election results can still be obtained with a relatively small number of checks. Additionally since the receipt does not contain information about how you voted, you can share this information with others, and allow them to perform this verification step on your behalf.

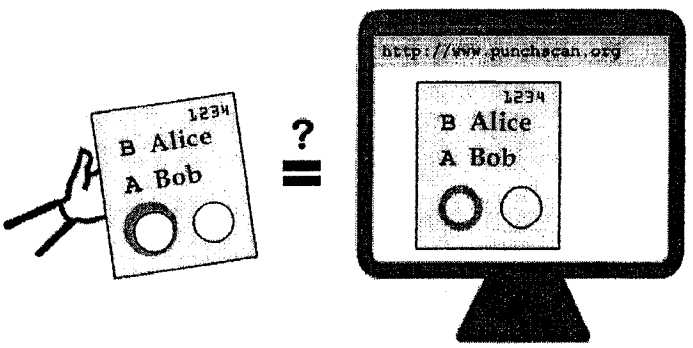

Figure 3.6: Online Receipt Check

Therefore receipt information can be published freely, for example in a newspaper. We have chosen instead to implement the receipt check as an online solution. The voter visits the election website, types in the serial number appearing on their ballot receipt. A diagram of their ballot is displayed allowing the voter to verify that what they hold in their hand matches what was used in the decryption/tally.

3.5 Audit Challenge Generator

In the previous section we introduced the independent verification components of the Punchscan election. In the pre and post election audits we indicated that challenges would be made to reveal certain portions of the Punchboard. To preserve ballot secrecy, we

cannot publically reveal the entire Punchboard for audit. Recall however the Punchboard was modularly designed so that portions of it can still be revealed without compromising ballot secrecy. Precisely which portions will be revealed is determined by an audit challenge. In order for the audits to be successful, the challenges must be issued in a way that is statistically unbiased. Clearly, fraudulently modified portions of the Punchboard could escape scrutiny if the audit challenges were issued in a way that intentionally avoided the affected areas. Therefore the critical design requirement of audit challenges is that **no one entity or group must be able to exert control over, or predict beforehand, the challenges that will be issued, with any advantage over any other entity or group**. Assuming an adversarial (i.e. zero-sum) relationship between candidates in an election, this could simply be achieved by each candidate each being offered a quota of ballots to select for auditing. However this may or may not result in statistically unbiased audit challenges if some candidates were to collude. A simple alternative to produce fair selections might be to flip coins. However this method is also subject to undue influence and is extremely inefficient for generating the large amounts of challenges required for our purposes. For our system we have designed and implemented a process that will generate audit challenge such that no one will know a-priori which selections will be made. The audit challenge generator serves the following purpose:

- **Pre-Election Challenge**: Given the total number of ballots in the election, randomly select serial numbers of half the ballots to be published (i.e., spoiled) and audited.

- **Post-Electon Challenge**: Given the serial numbers of the ballot receipts cast in the election, for a given ballot select either the left or right half of the decryption table to be published and audited.

If the election authority does not know *a priori* which ballots will be checked, it faces a high probability of getting caught if it publishes false commitments. The ballots to be selected for the audit should not be guessable with any advantage at the time the

commitments are published. Additionally the challenges themselves must be verifiable after the challenges are made. On first blush it would seem we would require some form of random process (e.g. rolling dice) to perform these challenges. However unless you trust the dice and are personally available to witness it being rolled, you cannot be fully satisfied the process was not manipulated to ensure a particular outcome. In the following chapter we describe our implementation of the audit challenge generator: we use stock indices to produce fair and observable challenges.

In summarization, this chapter has introduced the high level architecture of Punchscan and presented a description of the system components found in the three areas of concentration: election administration, voting and independent verification. Particular emphasis was placed on a design rationale for the Punchboard.

Chapter 4

Functional Specifications

This chapter discusses the functionality of the Punchscan software implementation and presents a cryptographic specification of how the Punchboard is generated and audited. We begin by discussing how ballots are designed and specified. We explain how the election master keys are created as well as how the software for such trusted operations can be verified by trustees. The latter half of this chapter discusses how the master keys are used to create the Punchboard and audit data, and concludes with a detailed specification of how the audit data is used to verify a Punchscan election.

4.1 Ballot Template Design

The layout of a Punchscan ballot is created using ballot template software, which can use almost any standard desktop publishing application. A ballot template primarily consists of two types of objects; static "background" objects (text and graphics) and special-purpose markers used to indicate the future placement of dynamic (varying from ballot to ballot) content. In the case of the election presented in chapter 6, *Microsoft Publisher* was used to create the ballot template. In addition to the ability to place basic static text and graphics objects on the ballot, the only requirement for the ballot template software is the ability to mark certain locations with simple colored shapes. The shapes function as positional markers and the ballot authoring software (described in the next section) will search for

the position of the various markers. Of these markers, disk shapes represent items that will appear on the top page, and ring shapes represent items that will appear on the bottom page. These positional markers are used to place four types of items on the ballot:

1. Ballot serial number (top and bottom pages).
2. Letters appearing beside candidate names (top page).
3. Letters appearing through the holes (bottom page).
4. Scanner alignment marks (top and bottom pages).

An example of a ballot template is shown in Figure 4.1.

4.2 Ballot Authoring Tool

The Ballot Authoring tool performs several functions centered around the manufacture, printing and scanning of ballots. Using the ballot template, the Ballot Authoring software tool searches the image looking for the location of the ring/disc shaped markers, and it uses them to create three files:

1. `Drill file`,
2. `Ballot Map`,
3. `Graphic Prints file`.

The `Drill File` is an industry standard XML data file that communicates to the machinist the precise position of the various holes that will be drilled in the ballot paper stock. The `Ballot Map` is similar in function to the `Drill File`, except that it is used by the polling place software to recognize and orient the location of marks made by voters on the optically scanned ballot images. Finally, the Authoring tool takes the XML formatted `Prints File` (an XML formatted file which contains the the unique information of each ballot) and uses the `Ballot Map` to place this data (in a human readable font) at the

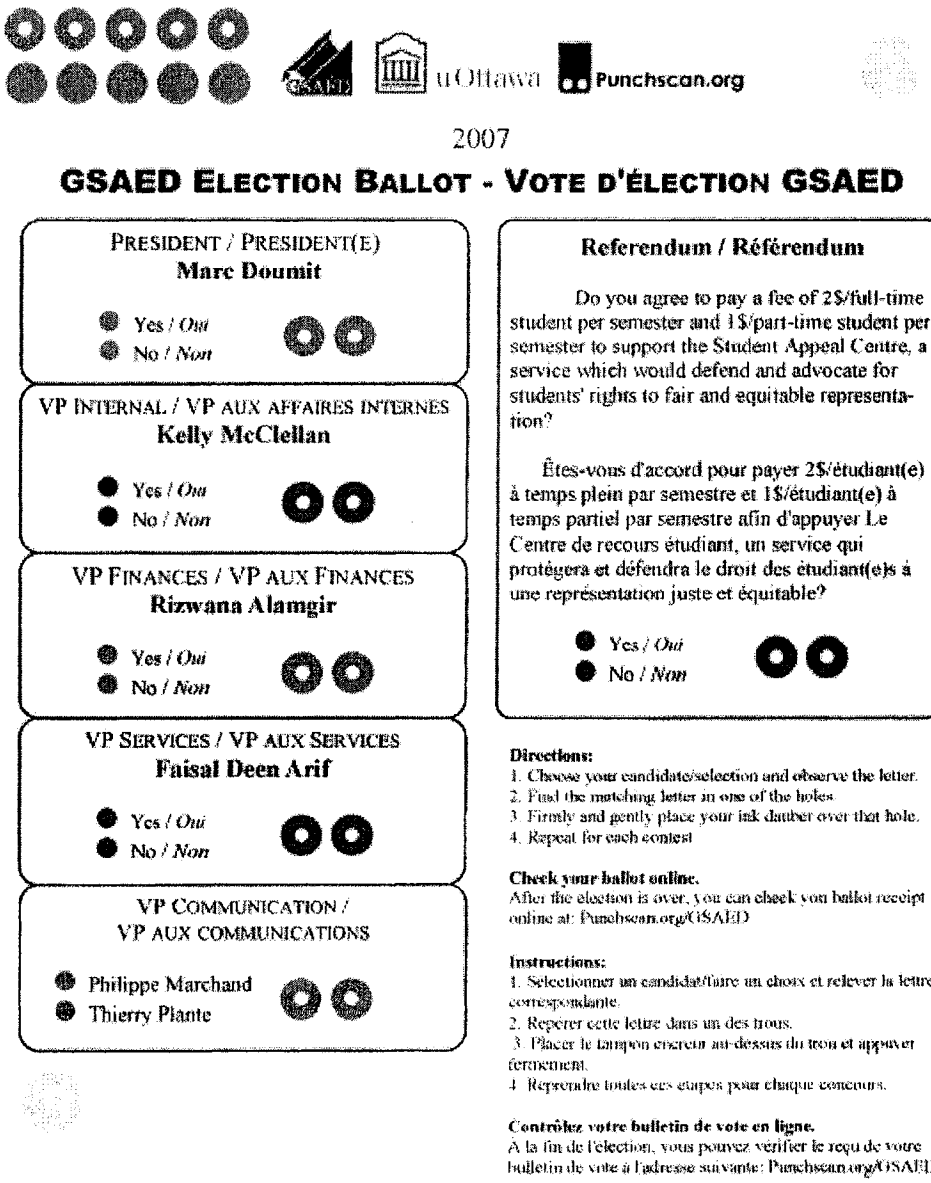

Figure 4.1: Ballot template used in the GSAÉD student election (described in chapter 6). Colored disk and donut shapes indicate the location where serial numbers and permutations will eventually be placed on the actual ballots.

specified locations on the ballot to create the graphical print-ready version of the ballot. The Authoring tool outputs this `Graphic Prints File` as a collection of PDFs, which will be used by the printer to print the paper ballots.

4.3 Election Engine

The election engine is the core software tool that performs the primary election administration functions of the trustees. It performs the following tasks:

- Generation of election master keys,
- Creation and population of the Punchboard,
- Creation of cryptographic commitments,
- Creation of "ballot print files,"
- Response to audit challenges,
- Decryption and tabulation of election results.

In addition it offers a validation tool that trustees can use to verify that the certified version of this software is the version being run at any given time.

4.3.1 Software Validation Tool

The election engine is always executed on a trusted hardware platform, in this case a diskless computer workstation. It is important to reliability and voter privacy that the version of the software being run throughout the election cannot be modified without detection. Each trustee is given the opportunity (before the election) to have the execution of the software validated or certified. Any time the trustees meet, it is important that it can be proven (by all trustees to all trustees) that the certified copy of the software is the precise software being executed. Under our implementation, each trustee is issued a copy of the certified software image on a removable storage device (e.g. USB thumb drive). In

addition to the engine itself, this software image includes a special bootable (aka "live") operating system, allowing the diskless workstation to be booted from a thumb drive and without the system requiring a hard drive. When the trustees meet, before the election engine software is run, the validation procedure is performed as follows:

1. Each trustee switches their bootable thumb drive to a read-only state.
2. One trustee's thumbdrive is selected to boot the diskless workstation from.
3. Once the system has booted, the other trustees (one at a time) place their thumb drive into the workstation. The currently executing version of the operating system hashes each respective thumb drive's software image and compares it to its own, ensuring they match.
4. Steps 2–3 are repeated until all trustees have verified each other's disk image.
5. One of these verified thumb drives is selected at random to be used as the software image for the remaining trustee tasks (described below).

Although this process has a complexity of $O(n^2)$, it is one means to ensure that any malicious or incorrect software cannot be executed undetected before sensitive data is entered into the system. In our implementation, the operating system[1] booted in about 2 minutes, and verification of one software image could be performed in about 1 minute. Therefore a three-trustee election would require about 15 minutes of verification before proceeding. In practice this is unlikely to prove an acceptable latency in the election workflow, and future work on this component is warranted. Nevertheless this scheme provides a reasonably simple and cost-effective measure for protecting privacy and ensuring proper operation of the system.

4.3.2 Election Master Key Creation

All of the Punchboard permutations and cryptographic commitments are generated from a pair of 128-bit keys refered to in this thesis as "election master keys." These keys are

[1] A scaled-down version of KNOPPIX (a Linux derivative).

a cryptographic combination of secret passphrases of the trustees. As a design requirement to mitigate service interruptions (due to absense, death, memory loss, etc) the keys can be regenerated by a pre-determined threshold of trustees. Furthermore the diskless workstation will ensure that no trustee is able to recover the key value–these values will only ever be present in the memory of the diskless workstation, which shall be purged at the conclusion of each meeting of the trustees. The keys are created from a cryptographic composition of the trustees usernames and secret passphrases.

Encrypted Password Files

All trustees are required to be present at the initial meeting. At future meetings only a pre-agreed threshold (i.e., subset) must be present to regenerate the election master keys. Each of the trustees enters a username and passphrase. During the first meeting, these credentials are created and stored (external to the diskless workstation) in a password database. Let $E_k(p)$ represent the symmetric encryption of p under key k. Let $H(q)$ represent the cryptographically-secure hash of q. Given c, a publically known election-specific constant, the format of the secret-sharing credential database for n trustees is:

$$\begin{aligned} Credential_{u_1} &= E_{H(u_1|p_1)}(c), \\ Credential_{u_2} &= E_{H(u_2|p_2)}(c), \\ &\vdots \\ Credential_{u_n} &= E_{H(u_n|p_n)}(c). \end{aligned}$$

During subsequent meetings, each trustee present enters their username/passphrase pair to regenerate their credential. This is checked against the corresponding credential in this database. If there is a mismatch, that username/password pair will not be included when regenerating the election master keys.

Key Generation and Threshold Regeneration

During the initial meeting of the n trustees (when all are present) the election master key S is generated from each trustee's username (u_i) and password (p_i) as follows:

$$S = H(u_1|u_2|\cdots|u_n|p_1|p_2|\cdots|p_n). \tag{4.1}$$

In our implementation, H represents the SHA-256 hash algorithm [46]. Two 128-bit keys are created by parsing the output $S = \{b_1, \cdots, b_{256}\}$ to create *two* masker keys defined as $Mk_1 = \{b_1, \cdots, b_{128}\}$ and $Mk_2 = \{b_{127}, \cdots, b_{256}\}$ where $b \in \{0, 1\}$. The notation Mk_1 and Mk_2 will be used in the cryptographic specification (see section 4.5), however for the remainder of this description, we will continue to refer to the keys as a single value, S. To enact the threshold feature, the key value S will be encrypted by various combinations of trustee username/password pairs, depending on both the overall number of trustees and the minimum threshold to regenerate the key. For n trustees with a threshold of k required to regenerate the master key, we encrypt S under a subset of username/passphrase combinations creating $\binom{n}{n-k}$ different threshold ciphertexts C as follows:

$$\begin{aligned}
C_1 &= E_{H(u_1|u_2|\cdots|u_k|p_1|p_2|\cdots|p_k)}(S), \\
C_2 &= E_{H(u_1|u_3|\cdots|u_{k+1}|p_1|p_3|\cdots|p_{k+1})}(S), \\
&\vdots \\
C_k &= E_{H(u_1|u_{n-k+1}|\cdots|u_n|p_1|p_{n-k+1}|\cdots|p_n)}(S), \\
C_{k+1} &= E_{H(u_2|u_3|\cdots|u_{k+1}|p_2|p_3|\cdots|p_{k+1})}(S), \\
&\vdots \\
C_{\binom{n}{n-k}} &= E_{H(u_{n-k+1}|u_{n-k+2}|\cdots|u_n|p_{n-k+1}|p_{n-k+2}|\cdots|p_n)}(S).
\end{aligned}$$

In the future, when a threshold of trustees enters their usernames and passwords, the corresponding ciphertext C_i is loaded from memory, and decrypted with the corresponding

combination of usernames and passphrases. Though this scheme has a poor space complexity, the number of trustees is likely to be small enough that the key generation and regeneration can be performed in reasonable time (i.e., imperceptible to the trustees). The system is however robust against error. Using the password files described earlier, incorrectly input information can be detected and attributed. In the event the encrypted password files are lost (including even the threshold ciphertexts), the regenerated key can still be validated. Any error will be immediately detected when regenerating the Punchboard–none of the cryptographic commitments will match their originally posted values.

4.4 Polling Place Software

The Polling Place software is constituted of the optical scan and record software used by the poll workers to record marks made by the voters. When the marked ballot receipts are scanned by an optical scanner, the polling place software will identify the location of the marks as well as the serial number of the ballot and store this information in two forms: the JPG image of the scan, and an XML formatted file of the detected marks. The former is retained for backup purposes with the latter constituting the data that will later be input into the Punchboard to recover the votes. More specifically, when a ballot is scanned the software will:

1. Search for alignment marks and apply a 2D inverse transformation (rotation/scale/translation) of the scanned ballot image to match the feature coordinates specified in the `BallotMap`,
2. Determine if the sheet scanned was a top or bottom sheet,
3. Use OCR (optical character recognition) to resolve the serial number,
4. Determine the location of voter-made marks,
5. Interpret the validity of mark placement, and present feedback to the pollworker. Overlays are displayed over marks. Illegally placed marks as well as unmarked regions are overlaid with red 'X' shapes, and green 'check marks' are made over valid marks.

6. Create an XML record of the (valid) mark locations.

7. Allow the pollworker to cast the ballot (and thereby store the XML record).

8. Print the aforementioned overlays onto the physical ballot receipt along with a digital signature barcode.

4.5 Cryptographic Specification

This section specifies the cryptographic algorithms used in creating the Punchboard and its commitments. These algorithms only employ symmetric-key encryption and cryptographic hash primitives, in contrast to the comparatively complex public-key approach of Prêt à Voter [11, 57, 66].

4.5.1 Ballot Keys

Each individual ballot requires its own set of "keys" which will later be used by the commitment function to "commit to" (encrypt/hash) the secret information contained in the ballot. Although these commitments are made public, the actual information contained in the ballot remains sealed. To create these ballot "keys," we use a identifier to function as "key material" and is denoted as Km_i. The key material uniquely identifies a particular instance of a particular parameter in the Punchboard. Typically this can represent the row number concatenated with the name of the parameter being committed to. Recall from our description of the Punchboard in section 3.3.1, for b instances of the decryption table, each ballot contributes 2 commitments to the P-table as well as 2 commitments to each of the b instances of the D-table. For example to commit to the left decryption/permutation of i-th row of the b-th instance of the decryption table, $Km_i = \{i||b||P_\beta D_L\}$. The election master keys Mk_1 and Mk_2 are generated by the trustees during their first meeting. There is also an election constant C which protects the election master keys across different elections, should trustee passwords ever reused for whatever reason.

Algorithm 1: Ballot Key

Input: Km_i, Mk_1, Mk_2, C
Output: Skm
1 $\text{Skm}_i = D_{Mk_1}(C \oplus E_{Mk_2}(C \oplus E_{Mk_1}(Km_i)))$

In our implementation, the encryption/decryption functions listed above are performed using AES [19] with a 128-bit key size. The ballot key generated from the key material, election master keys, and election constant is used as a "salt" to the commitment function, and is referred to here as "salt with key material," (i.e. Skm).

4.5.2 Commitments

The commitment function uses the AES block cipher with 128-bit key encryption in electronic codebook (ECB) cipher mode, (denoted below as `AES128`). Additionally it uses the SHA-256 hash algorithm denoted below as `SHA256`. The inputs and outputs to this function are base64 encoded so that they may reside in ASCII form in the corresponding XML election data files. However, the cryptographic operations below are preformed at the byte level. Each ballot i has four pieces of information M that are to be committed to; the permutations on the top and bottom sheets (P1, P2) as well as the left and right decryption permutations (D2, D4). Each item within a ballot is committed to separately. The message being committed to (the value of either P1, P2, D1/D2 or D4/D5) is denoted as MX, and the commitment of MX denoted as CX.

Algorithm 2: Commitment

Input: MX_i, Skm_i, C
Output: CX_i
1 $Sak_i = \texttt{AES128}_{Skm_i}(C)$
2 $h1_i = \texttt{SHA256}(MX_i || Sak_i)$
3 $h2_i = \texttt{SHA256}(MX_i || \texttt{AES128}_{Sak_i}(h1_i))$
4 $\text{CX}_i = h1_i || h2_i$

4.5.3 Punchboard Permutations

At each meeting of the trustees the Punchboard is regenerated in the memory of the diskless workstation. Recall that the Punchboard has a "ballot symbol permutation" as well as a pair of row permutations for each ballot entry. Recall from section 3.3.1 that the Punchboard permutes all rows between the P- and D-, as well as between the D- and R-tables in a "random-looking" way. For an election with b ballots, we require a scheme that can generate two (cryptographically distinct) permutations on b elements. Although there are other more efficient means to generate permutations, the algorithm used in our implementation can be summarized simply as encrypting the elements of an identity permutation and then sorting them to produce a "random-looking" permutation.

Algorithm 3: Row Permutation

Input: Mk
Output: Row Permutation p

1 **for** $i = 1 \cdots b$ **do**
2 $\quad T[i][1] \leftarrow i$
3 $\quad T[i][2] \leftarrow E_{Mk}(i)$
4 `Sort`(T) by $T[*][2]$
5 **for** $i = 1 \cdots b$ **do**
6 $\quad p(i) \leftarrow T[i][1]$

The algorithm used to generate the pseudo-random permutation of the symbols contained on each ballot is very similar to the Row Permutation algorithm, except requires the unique ballot key and generates the (statistically independent) permutations for the top and bottom sheets.

4.6 Audit Data Specification

In this section we define the audit data and provide the chronology of its generation and use.

Algorithm 4: Ballot Symbol Permutations

Input: Ballot Symbol Set $s = \{a, b, \cdots, n_{candidates}\}, Skm_i$

Output: Symbol Permutations (for ballot j), $TopSheet_j, BottomSheet_j$

```
for i = 1 ··· n_candidates do
    T[i][1], B[i][1] ← s(j)
    T[i][2] ← E_{H(Skm_i||"Top")}(i)
    B[i][2] ← E_{H(Skm_i||"Bottom")}(i)
Sort(T) by T[*][2]
Sort(B) by B[*][2]
for j = 1 ··· n_candidates do
    TopSheet_j(i) ← T[i][1]
    BottomSheet_j(i) ← B[i][1]
```

4.6.1 Definition of Terms

Before we can begin explaining the audit process, we need to return to the Punchboard structure and present a more systematic naming convention used in the software implementation. Recall there are 3 main columns; "prints" table (**P-Table**), the "decryption" table (**D-Table**) and the results table (**R-Table**). Each ballot i forms one complete row in the Punchboard, but the row number of ballot i's entry differs in the P, D and R tables, as governed by a row permutation. Let ballot i's entry in the Punchboard be located on rows p, q, r of the P-, D- and R-tables respectively. We introduce this notation in table 4.1.

P-Table	
$\texttt{P1}_i$	permutation on the top page of ballot i
$\texttt{P2}_i$	permutation on the bottom page of ballot i
$\texttt{P3}_i$	position of the voter's mark on ballot i
$\texttt{CP1}_i$	commitment to P1 (in row p) of ballot i
$\texttt{CP2}_i$	commitment to P2 (in row p) of ballot i
D-Table	
$\texttt{D1}_i$	pointer to row p in the P-Table
$\texttt{D2}_i$	first decryption permutation of ballot i
$\texttt{D3}_i$	partially decrypted vote on of ballot i
$\texttt{D4}_i$	second decryption permutation of ballot i
$\texttt{D5}_i$	pointer to row r in the R-table
$\texttt{CD2}_i$	commitment to D2, D1 (in row r) of ballot i
$\texttt{CD4}_i$	commitment to D4, D5 (in row r) of ballot i
R-Table	
$\texttt{R}_i$	result (i.e. "unencrypted" vote) of ballot i

Table 4.1: Punchboard Notation

4.6.2 Data Files Used and Produced by the Meetings of the Trustees

During the meetings of the trustees, election and audit data is both used and produced by meetings of the election trustees. Recall there are four meetings of the trustees, and each meeting has an input and output associated with it. The data is contained in XML files that take the name associated with its meeting and function. The meetings files are labeled `Meeting{One, Two, Three, Four}{In, Out}`

Meeting 1 (First Meeting of the Trustees)

During their fist meeting, the trustees instantiate the Punchboard using the election master key. During this time `P1`, `P2`, `D1`, `D2`, `D4`, and `D5` are generated, but kept secret. The election engine then, for each ballot, computes and returns the commitments `CP1`, `CP2`, `CD2` and `CD4` of `P1`, `P2`, `D2`, and `D4` respectively.

- `MeetingOneIn` specifies how many ballots to generate n, how many instances b of the D-table to generate, as well as the election constant C. Additionally it contains a file pointer to the election specification file which contains the number of contests, as well as the number of respective candidates. This information determines the size of the algebraic group G that will be used to construct permutations and their inverses.
- `MeetingOneOut` contains the commitments `CP1`, `CP2`, `CD2` and `CD4` of all n ballots.

Meeting 2 (Public Disclosure of Pre-Election Audit Data)

During the pre-election audit challenge, the serial numbers of half of the n ballots are selected (randomly) to be "unsealed" and made public for auditing purposes. In meeting 2, the trustees use their passwords to regenerate the Punchboard, and output the secret information of the requested ballots. The ballots are now considered "spoiled" and will not be physically printed or voted on in the election.

- `MeetingTwoIn` contains the list of $n/2$ serial numbers to be audited

- `MeetingTwoOut` returns `P1`, `P2`, and each of the b instances of `D1`, `D2`, `D3`, `D4` and `D5` for the selected ballots

- `Prints File` contains `P1` and `P2` of all remaining (unaudited) ballots. This file is intended to be seen only by the trusted printing authority for the purposes of printing the ballots.

Meeting 3 (Ballot Decryption and Election Tally)

As the election wraps up and the polling period ends, the ballot sheets scanned at the polling places throughout the election are collected together, decrypted and tallied. Recall when a voter casts their vote, they destroy one sheet of their ballot, and present the other half to be scanned at the polling station. This ballot receipt contains the ballot's serial number i, the position of the mark the voter made (`P3`) and which sheet they chose to retain (either top or bottom). Once again the trustees enter their passwords, the Punchboard is recreated. The polling place data is entered, and the results are generated. Along with the voting results, meeting 3 also outputs the permutation on the sheet that was retained by the voter (to audit the printing), as well as the intermediate "partially decrypted" vote to allow for auditing of the D-Tables.

- `MeetingThreeIn` essentially contains all the scanned ballot receipts (i.e. encrypted ballots) cast in the election. Specifically, it contains a record of either "top" or "bottom" as having been retained by the voter as well as the mark position ($\texttt{P3}_i$) made by the voter for each ballot i cast in the election.

- `MeetingThreeOut` returns (for each of the n ballots) `R`, b instances of `D3` and *one* of either $\texttt{P1}_i$ or $\texttt{P2}_i$ depending if the voter retained the top or bottom sheet respectively.

Meeting 4 (Public Disclosure of Post-Election Audit Data)

Similar to `meeting` 2, this meeting also responds to an audit challenge. However this time, we cannot simultaneously reveal permutations `D1` and `D5`. This would link serial numbers

to their respective entry in the R table (i.e. the vote). Additionally we cannot reveal D2 and D4 for similar reasons. For each ballot, the post-election audit challenge requests either the left side of the decryption table D1, D2 or right side D4, D5 for each of the b instances thereof.

- MeetingFourIn contains the side of the decryption table to open (left or right) for each ballot.
- MeetingFourOut returns the corresponding tuple, either (D1, D2) or (D4, D5) for each ballot.

Now that we have covered the location and timeline for generating the audit data, we can begin discussing the specific checks made against this data during the audit.

4.7 Election Audit Specification

In this section we explain the details behind how Punchscan performs these audits. Broadly speaking, an audit of an E2E voting system should, at minimum, undertake to allow voters to independently verify the following:

1. The ballot data requested for audit by the verifier was indeed the ballot data received from the election administration.
2. The ballot data is in accord with its previously posted commitment, meaning that the election administration did not alter the ballot given that it was requested for audit.
3. The ballot data is correctly formatted[2].
4. The number of encrypted ballots input to the decryption function matches the number of decrypted ballots output from the decryption function.

[2]This obviously differs from system to system, however in an E2E system, it would likely refer to verifying the correctness of some mathematical relationship (e.g., does 1+1 = 2?)

5. The information contained on the ballot receipts retained by voters was indeed the information input put through the decryption function.

Items 1–4 are verified by the pre and post election audit tools, and item 5 is verified by the online receipt checker. It is required in a Punchscan election that any interested citizen or group be able to personally perform the audit to independently verify its findings. Therefore we must design a system based on notion of availability. The audit must be able to be independently (and repeatably) performed by any interested party on the platform of their choosing. There are four tools that we need to make available for voters to perform the audits. They are:

1. Audit Challenge Generator
2. Pre-Election Audit Tool
3. Post-Election Audit Tool
4. Online Receipt Checker

First is the audit challenge generator which fairly selects ballots to be audited. This could be done by flipping a coin to decide if a given ballot should be revealed and audited. In our implementation we use stock indices to make these selections in an unbiased, unpredictable but widely verifiable way. Now that we have a means to fairly challenge the election administration for ballot data, we also require software tools to perform the cryptographic integrity checks of the pre- and post-election audits. Finally voters need a means to verify that the information contained on their ballot receipt was actually used in the decyption/tally process.

4.7.1 Audit Challenge Generators

One of the essential requirements of Punchscan is a secure source of randomness, the results of which cannot be directly known a priori, or deterministically influenced by any entity with any advantage. This could be achieved, for example, by using a set of unbiased dice.

However proving the fairness, and observing the rolling of dice may not be practical in all situations. As an alternative Clark and Essex propose a scheme in [17] that uses stock indices as a source of secure randomness.

Stock data is well suited for this task because based on the assumption that future closing price of a stock is unpredictable[3] *a priori*, yet easily verified *a posteriori*. For each stock in the portfolio, the closing price is concatenated into a long integer. This integer is then used as a seed to a pseudorandom number generator (PRNG). The PRNG does not need to be cryptographically secure because the seed is public knowledge after the stock market closes on the specified date. Our system uses a certain non-linearly evolving cellular automata, which is known to have good random properties and adhere to the strict avalanche criterion [17]. The stock portfolio contains 32 stocks—the subset of the *Wired 40* companies [53] which are traded on NASDAQ. The statistical properties of using stocks in this fashion, and their relationship to auditing in Punchscan, has been studied and found secure [17]. The audit challenge generators (using stock data) for both the pre- and post-election audits are shown in the `Pre/Post Election Audit Challenge Generator` algorithms below.

Algorithm 5: Pre-Election Audit Challenge Generator

Input: `MeetingOneIn`, Pre-election Stock Portfolio data
Output: `MeetingTwoIn`

$n \leftarrow$ `MeetingOneIn.noBallots`
$seed \leftarrow \emptyset$
foreach *stock* $s_i \in$ *Pre-Election Portfolio Data* **do**
    $p_i \leftarrow$ `ClosingPrice(`s_i, *timestamp*`)`
    $seed \leftarrow seed || p_i$
selection bit stream **b** $\leftarrow$`PRNG` (seed)$\in \{0,1\}^n$
for $i = 1 \cdots n$ **do**
    if $b_i \in \{0,1\} = 0$ **then**
        Add serial number i to `MeetingTwoIn`

[3]The precise entropy of a stock index is difficult to determine on average, however we reasonably assume any stock index to provide at minimum one bit of entropy across a full trading day.

Algorithm 6: Post-Election Audit Challenge Generator

Input: `MeetingThreeIn`, Post-election Stock Portfolio data
Output: `MeetingFourIn`

```
1  n ← Set of all serial numbers in MeetingThreeIn
2  seed ← ∅
3  foreach stock s_i ∈ Pre-Election Portfolio Data do
4  |  p_i ← ClosingPrice(s_i, timestamp)
5  |  seed ← seed||p_i
6  selection bit stream b ←PRNG (seed)∈ {0,1}^|n|
7  for i = 1 ··· |n| do
8  |  if b_i ∈ {0,1} = 0 then
9  |  |  Add serial number n_i and "Left" to MeetingFourIn
10 |  else
11 |  |  Add serial number n_i and "Right" to MeetingFourIn
```

4.7.2 Pre-Election Audit Tool

Recall at the beginning of this section we explained the requirements for an E2E audit. Relevant to the pre-election audit, we must verify that:

1. The ballot data requested for audit by the verifier was indeed the ballot data received from the election administration.

2. The ballot data is in accord with its previously posted commitment, meaning that the election administration did not alter the ballot given that it was requested for audit.

3. The ballot data is correctly formatted.

See the following `Pre-Election Audit` algorithm. Lines 2–3 verify item 1 of the above audit criteria. Lines 4–7 verify item 2 of the audit criteria. Finally lines 8–9 verify item 3 of the audit criteria.

4.7.3 Post-Election Audit Tool

Relevant to the post-election audit, we must verify that:

Algorithm 7: Pre-Election Audit

Input: MeetingOneIn, MeetingOneOut, MeetingTwoIn, MeetingTwoOut
Output: Pre-Election Audit Report

```
1 foreach i ∈ MeetingTwoIn do
2     if i ∉ MeetingTwoOut then
3         Report missing ballot i
4     if Commitment(P1_i) ≠ CP1_i then
5         Report invalid P1_i and/or CP1_i
6     if Commitment(P2_i) ≠ CP2_i then
7         Report invalid P2_i and/or CP2_i
8     if (P2_i ∘ P1_i) ≠ (D4_i ∘ D2_i)^-1 then
9         Report invalid (P1,P2,D2,D4)_i
```

1. The ballot data requested for audit by the verifier was indeed the ballot data received from the election administration.

2. The ballot data is in accord with its previously posted commitment, meaning that the election administration did not alter the ballot given that it was requested for audit.

3. The ballot data is correctly formatted[4].

4. The number of encrypted ballots input to the decryption function matches the number of decrypted ballots output from the decryption function.

See the following `Post-Election Audit` algorithm. Lines 11-12 verify item 1 of the above audit criteria. Lines 3–9 verify item 2 of the audit criteria. Lines 13–22 verify item 3 of the audit criteria. Finally, lines 1–2 verify item 4.

4.7.4 Online Receipt Checker

This aspect of independent verification is different from the others in that the two items being compared are in different mediums. In the pre- and post-election audits, we are computing and comparing digital bits. However in this circumstance we are comparing

[4]This obviously differs from system to system, however in an E2E system, it would likely refer to verifying the correctness of some mathematical relationship (e.g., does 1+1 = 2?)

Algorithm 8: Post-Election Audit

Input: `MeetingOneIn`, `MeetingOneOut`, `MeetingThreeIn`, `MeetingThreeOut`, `MeetingFourIn`, `MeetingFourOut`

Output: Post-Election Audit Report

```
1  if |ballots| in MeetingThreeIn ≠ |ballots| in MeetingThreeOut then
2      Report ballot count error
3  foreach Receipt_i ∈ MeetingThreeIn do
4      if Receipt_i = top sheet then
5          PX_i = P1_i, CPX_i = CP1_i
6      else if Receipt_i = bottom sheet then
7          PX_i = P2_i, CPX_i = CP2_i
8      if Commitment(PX_i) ≠ CPX_i then
9          Report invalid PX_i and/or CPX_i
10 foreach side s_i (left or right) ∈ MeetingFourIn and side s_i* ∈ MeetingFourOut do
11     if s_i ≠ s_i* then
12         Report challenge mismatch at i
13     if s_i = Left then
14         DX_i = D2_i, CDX_i = CD2_i
15         if D3 ≠ D2(P3) then
16             Report invalid decryption
17     else if s_i = Right then
18         DX_i = D4_i, CDX_i = CD4_i
19         if R ≠ D5(D3) then
20             Report invalid decryption
21     if Commitment(DX_i) ≠ CDX_i then
22         Report DX_i, CDX_i
```

the information contained on a physical object with its electronic manifestation. Recall, in this step the voter checks to see that what they hold in their hand matches what went into the decryption/tally. The algorithm below describes the verification process the voter undertakes. It is important to realize however that the information is displayed in a graphical way, and that although the notation below may seem complicated, the human auditor *is* reasonably expected to be able to notice any inconsistencies. See the image below.

Algorithm 9: Online Receipt Check

Input: Physical Ballot: Serial number s^*, P3* and one of $\{P1^*, P2^*\}$ (permutation appearing on either the top or bottom sheet) respectively

```
1  {s, P3, PX} ← ReceiptCheckingWebsite(s*), where PX = one of {P1, P2}
2  if s* ≠ s then
3  |  Serial number mismatch
4  if P3* ≠ P3 then
5  |  Mark transcription error
6  if Voter kept top sheet then
7  |  if PX = P1 then
8  |  |  if PX ≠ P1* then
9  |  |  |  Print Error
10 |  else
11 |  |  Wrong sheet recorded
12 if Voter kept bottom sheet then
13 |  if PX = P2 then
14 |  |  if PX ≠ P2* then
15 |  |  |  Print Error
16 |  else
17 |  |  Wrong sheet recorded
```

This chapter has examined some of the implementation specific aspects of Punchscan. The first half of the chapter outlined how ballots are designed and generated, how the election engine software is verified and how the election master keys are generated. We then turned our attention to the cryptographic aspects of the system by describing how the Punchboard and commitments are created. Finally a specification of the audit data and algorithms outlined how a Punchscan election is verified.

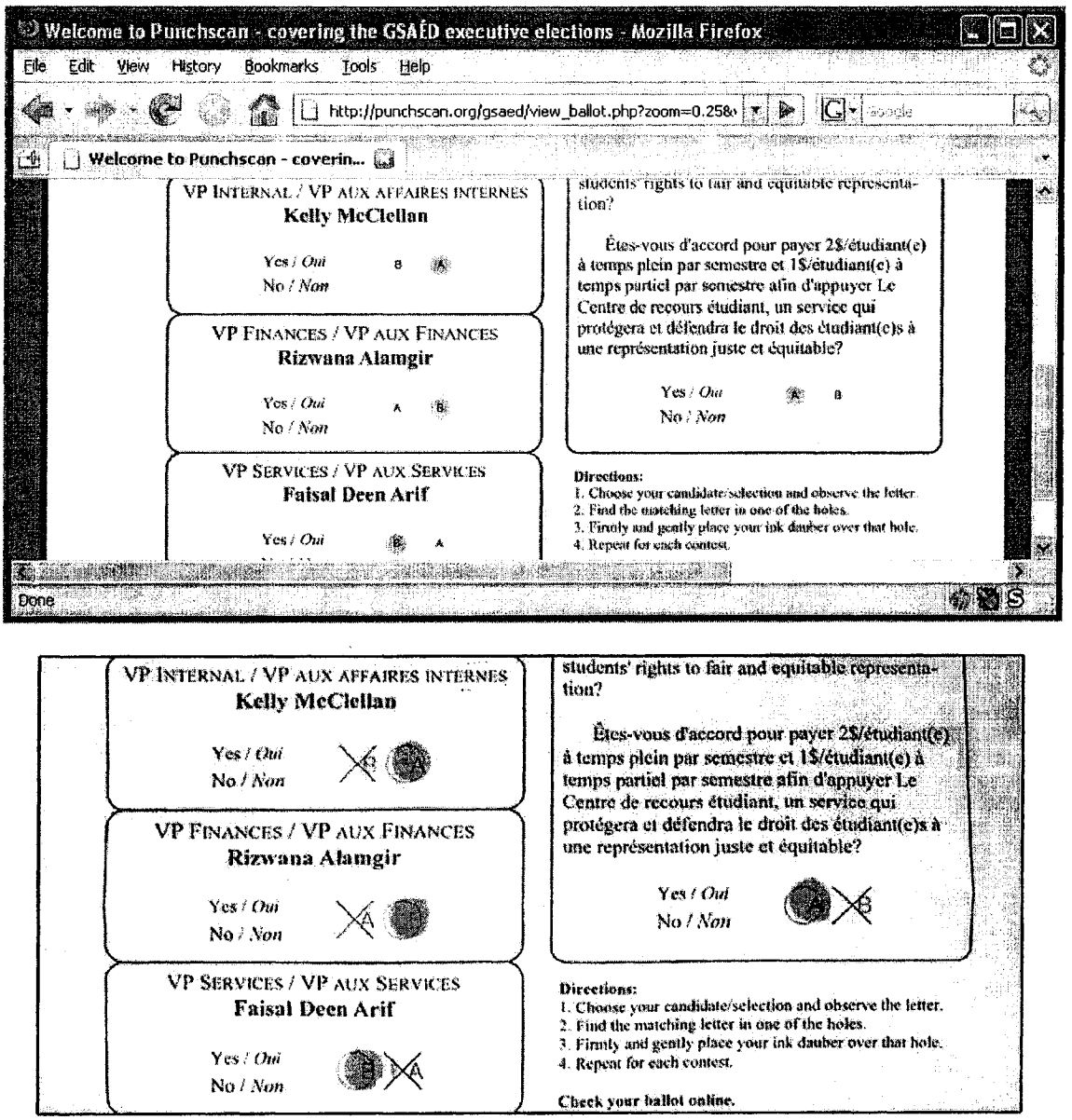

Figure 4.2: (**Top**) Online receipt display tool showing ballot #04078 from the 2007 uOttawa GSAED election. (**Below**) Photograph of the paper original showing voter made ink daubs (orange) and printed overlays.

Chapter 5

Election Procedures

In this chapter we present the timeline and procedures undertaken in a Punchscan election. In addition to the procedures one might expect in a conventional paper ballot or optical scan voting system, an E2E election has additional procedural elements surrounding the verification of the results. The election process is divided into three main time periods: pre-election, election and post-election phases. A Gantt chart of the election timeline is shown in Figure 5.1.

Recal from chapter 3 there are four categories of user: election trustees, voters, poll workers and independent verifiers. The following list presents the order of events of a Punchscan election indicating which type of user(s) is involved in each event:

- Election specification (*Trustees*),
- Ballot template design (*Trustees*),
- Ballot manufacture (*Trustees*),
- `Meeting 1` (*Trustees*),
- Ballot printing (*Trustees*),
- Pre-election challenge (*Trustees and independent verifiers*),
- `Meeting 2` (*Trustees*),

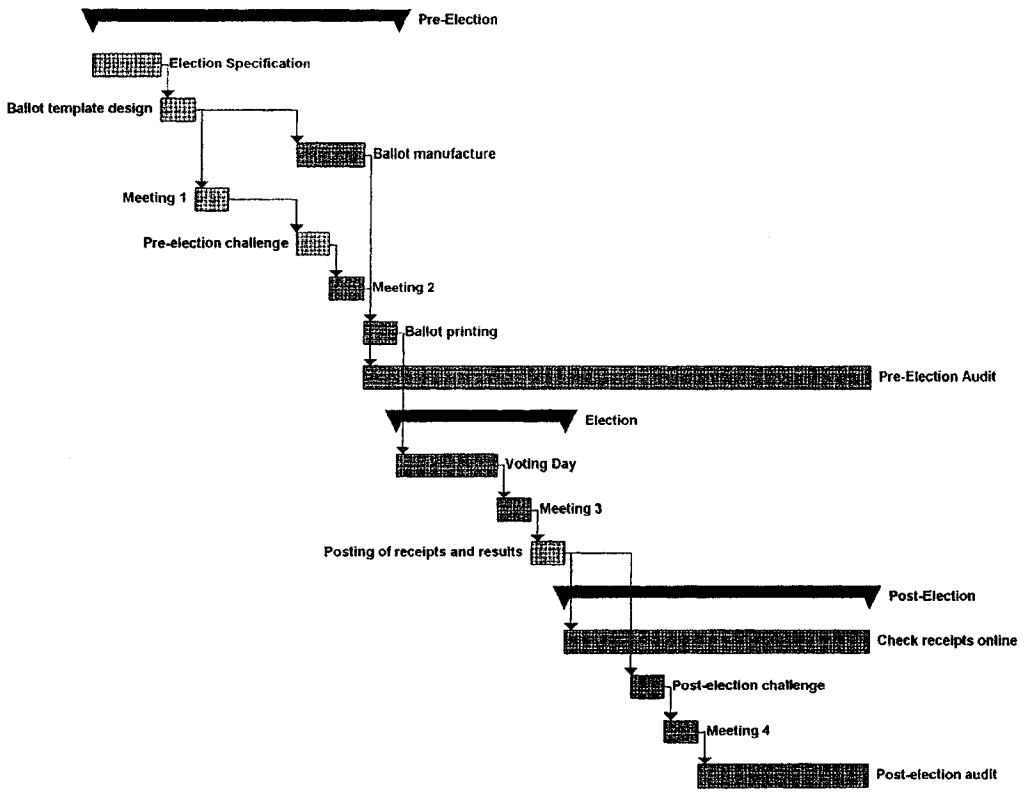

Figure 5.1: **Punchscan election timeline (Gantt Chart).** Task durations are relative. Generally speaking, the meetings of the trustees, audits and audit challenges can be performed on the order of minutes. The duration of the election specification phase, ballot template design and voting days are election specific, and are determined by the needs of the electoral body. Ballot manufacture and printing are on the order of a day, depending on logistics. The lag time of the pre and post-election audit challenges are prescribed to be one business day at minimum. The date up to which voters may perform independent verification via the audits and online receipt checking (and potentially contest the election results) will be mandated by pre-existing election policy/law.

- Pre-election audit (*Independent verifiers*),
- Voting day(s), (*Voters and poll workers*),
- `Meeting 3` (*Trustees*),
- Posting of receipts and results (*Trustees*),
- Check receipts online (*Voters*),
- Post-election challenge (*Trustees and independent verifiers*),
- `Meeting 4` (*Trustees*),
- Post-election audit (*Independent verifiers*).

In the following sections we will expand on these procedures from the perspective of the trustees, voters, poll workers and independent verifiers respectively.

5.1 Trustee Procedure

Recall there are four meetings of the election trustees as well as an initial election definition phase. At each meeting, the trustees boot the diskless workstation from a removable storage medium (with the election engine software and the live booting operating system). Each trustee will validate the others' instances of the software and enter their passphrases (as described in chapter 4), which will regenerate the state of the Punchboard in memory.

5.1.1 Election Definition Phase

During this preliminary phase the ballot template is designed in accordance with the election rules and list of candidates. The Ballot Authoring Software is run to generate the drill coordinates file and this file is given to the machinist to manufacture (i.e. drill) the ballot paper stock. During `Meeting 1` the passphrases are entered for the first time to generate threshold/password files, and create the election master keys. The Punchboard

is also generated for the first time, and the corresponding cryptographic commitments are computed, output and posted on the public election bulletin board (i.e. webserver).

5.1.2 Pre-Election Phase

Once the Punchboard commitments to Punchboard are published a waiting period begins, the conclusion of which sees the audit-challenge generation tool generate the pre-election challenges. `Meeting 2` is held and the secret data of challenged ballots are output to the bulletin board for audit. The remaining secret ballot data is run through the Ballot Authoring software and the Graphic Prints file is created and given to the ballot printers. The printed paper ballots are transported in sealed envelopes to the polling location in preparation for the election.

5.1.3 Election Phase

The trustees are not involved in the election phase until the very end. Once all the polling stations have reported the scan data (marks made by voters), `Meeting 3` is held. The Punchboard is regenerated in memory, and the scan data is entered into the election engine. The program runs the mark data through the Punchboard to produce the "unencrypted" (and unlinkable) versions of voters' votes. This data is aggregated (i.e. counted) to produce the election tally, and the results are posted to the bulletin board.

5.1.4 Post-Election Phase

Once the election results are published another waiting period begins, the conclusion of which sees the audit-challenge generation tool generate the post-election challenges. `Meeting 4` is held and the secret data of the challenged decryption keys are output to the bulletin board for audit.

5.2 Voter Procedure

To mark and cast a ballot the voter performs the following steps at the polling place:

1. State the intention to keep either the top or bottom page,
2. Receive a ballot and enter the polling booth,
3. Open the clipboard assembly to view the ballot,
4. For each contest:
   (a) Decide on and locate candidate/option to vote for,
   (b) Note the symbol appearing beside that candidate/option,
   (c) Locate the hole containing the same symbol;,
   (d) Mark the hole by firmly daubing it,
5. In view of the poll workers, remove and shred either the top or bottom sheet (as stated in 1),
6. Return the clipboard with the remaining sheet (still locked) to poll workers,
7. Check that the scanned image captures the mark positions and if so direct the poll worker to cast (i.e. record into memory) the ballot,
8. Obtain receipt.

After leaving the polling place, the voter is free to distribute copies of their receipt to voting organizations for the purposes checking the receipt on voter's behalf. Whether or not the voting procedures described above would meet a universal standard of usability has not been definitively established and is a subject of future work.

5.3 Poll Worker Procedure

When a voter enters the polling station to vote, the poll workers perform the following steps to prepare a ballot to be voted on:

1. Look up the voter in the voter list,
2. Place the top sheet of a new ballot face down on the clipboard assembly,
3. Place the bottom sheet face down on the clipboard assembly,
4. Close the clipboard assembly so that ballot marks are covered and only the ballot serial numbers are showing,
5. Check that the serial numbers on the top and bottom sheet match,
6. Ask the voter which layer they intend to keep,
7. Lock the ballot in place and give the clipboard to the voter,
8. Provide instructions on how to mark the ballot.

Once the voter has returned with a marked ballot the poll worker records their marks as follows:

1. Unlock the clipboard and remove the remaining ballot sheet,
2. Scan the ballot sheet and display the results of the OCR to the voter,
3. If approved by the voter, cast the ballot,
4. Indicate the voter has *voted* in the voter list,
5. Place the sheet in the printer for printing the overlays, digital signature, and paper-based backup copy,
6. Cut the paper chit off the bottom corner of the sheet,

7. Return the sheet to the voter to be kept as a receipt,

8. Place the paper chit in an envelope for safekeeping.

If the voter returns with a ballot that has been removed from the lock or a ballot on which they have made a mistake, they can be reissued a ballot following the same procedure. Typically the number of attempts will be bounded. Whether or not poll workers will require a degree of training exceeding that of an existing optical scan/DRE election has not been definitively established and is a subject of future work.

5.4 Independent Verifier Procedure

The independent verifiers are given several means to participate in the verification process: the audit challenge verification (to ensure the pre/post election audit data is disclosed in an unbiased manner), the pre/post election audits (to ensure counted-as-recorded integrity), the receipt check (to ensure recorded-as-marked integrity). Each verification step is voluntary.

Pre/Post-election audit challenge verification To verify the audit challenges are made in a fair (i.e. verifiably random) way, the verifier reproduces the challenges using the appropriate day's stock market data. A software tool was written to automate the following steps:

1. Download appropriate stock market data,

2. Run audit challenge generator to produce challenges,

3. Compare these challenges to the officially stated challenges and check for discrepancies.

Pre/Post-election Audit To verify recorded marks will be/were decrypted correctly, the verifier checks the audit data conforms to the cryptographic relationship outlined in chapter 4. A software audit tool was written to automate the following steps:

1. Download the appropriate audit data (challenged data and their original commitments),

2. Run the audit tool to generate commitments of the challenged data,

3. Compare these commitments to the originally posted versions and check for discrepancies.

Receipt Check To verify the marks made on the ballots were recorded correctly, the verifier (or another verifier on their behalf) checks the paper receipt against the officially reported version. An online tool (shown in figure 4.2) was written to display the XML data in a graphical way. The officially reported marks are verified through the following steps:

1. Visit the election website.

2. Type in ballot serial number.

3. Check that the marks displayed on the online version match those on the paper receipt.

The procedures presented in this chapter offer a brief but complete account of how a Punchscan election is run and verified.

Chapter 6

Campus Election Results

In this chapter we present a case study of the first use of Punchscan (and an E2E system) in an election with binding results. The system components as described in chapters 3 and 4, as well as procedures outlined in chapter 5 are applied here to running a real-world election. The results presented here represent a critical step in the design of Punchscan offering both a solid proof-of-concept and an initial sense of how well election participants can assimilate the extra facets of independent verification.

The University of Ottawa's *Graduate Students' Association / Association des étudiant(e)s diplômé(e)s* (GSAÉD) voted unanimously to adopt the Punchscan voting system for their 2007 executive election held over the three day period of March 6–8. Among the reasons they cited for their decision was the desire to speed up the tally process, increase the integrity of election results, provide a means to identify double-voting, and, at an academic level, play a leading role in voting systems research.[1]

The GSAÉD's chief returning officer (CRO) in conjunction with the Punchscan team proceeded to conduct the first binding election with independently verifiable results.

[1]This chapter reprinted in large part from [27].

Figure 6.1: A pollworker explains to a voter how to mark a Punchscan ballot during the GSAÉD election in Ottawa.

6.1 Election Requirements

In consultation with GSAÉD, several requirements for this election were arrived upon. GSAÉD required there to be five polling stations located on the University of Ottawa campus, which were to be operated during normal business hours across a three day period. Two poll workers were to be present at all times. Being a fully bilingual university, the ballot was required to be worded in both English and French, and it was decided that this would be accomplished through a single bilingual ballot, as opposed to separate ballots for each language. The offices, candidate names, as well as French translations were provided by GSAÉD.

In addition to these basic requirements, it was agreed that several other additions over the basic Punchscan system would be implemented to provide greater security and robustness. They are as follows:

6.1.1 Ballot Receipt Digital Signatures

An E2E ballot receipt allows a voter to verify their vote was counted correctly, but also serves as a proof when it was not. At it's core it protects the voter from mistakes or malicious activity on the part of the election trustees. However given that Punchscan ballots are produced using low cost drilling and printing methods, fabricating a counterfeit receipt for the purposes of invalidating the election results is entirely feasible. Therefore we need a means to protect the election trustees (and in turn the election results) from malicious voters. This threat was addressed by the introduction of a barcode-based digital

signature printed onto the ballot receipt at the time that it was cast.

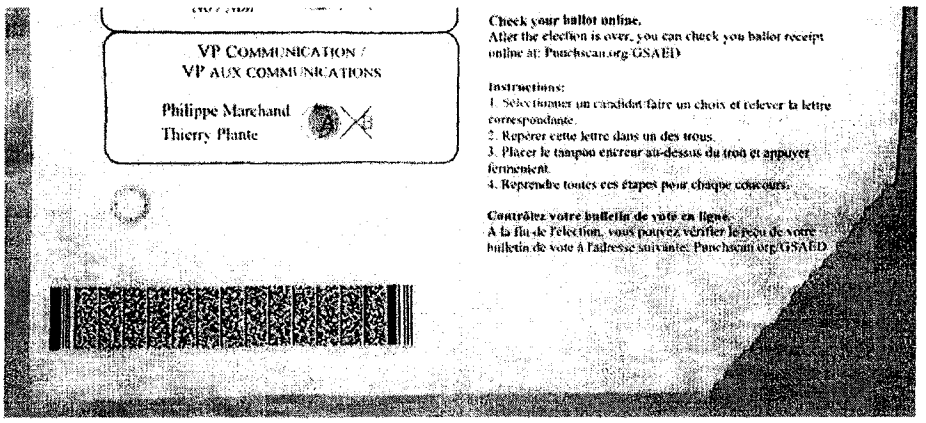

Figure 6.2: The bottom portion of a (GSAÉD) Punchscan ballot. (**Bottom left**) A 2D barcode representing a digital signature of the mark positions. (**Middle left**) Yellow alignment mark. (**Bottom right**) Cut line of the paper-based backup chit.

At the polling place, the ballot receipt is scanned and the Polling Place software detects and encodes the serial and mark positions into an XML file. The ballot receipt is then placed in a printer and green 'O' shaped overlays are printed over top of letters marked by ink daubs to confirm and 'lock-in' the location of the daub mark. Additionally 'X' shaped overlays are printed over unmarked positions. The marked positions are then digitally signed and printed on to the bottom left corner of the ballot receipt in a scanable barcode format, allowing the election trustees to establish the authenticity of a ballot receipt in the event of a future dispute.

6.1.2 Paper-based Backup

Punchscan records ballot marks by optically scanning and electronically storing the marks. Until this point however, GSAÉD had only ever employed traditional (i.e. strictly paper) ballot and had procedures for their handling, counting, and eventual disposal. Because Punchscan is an optical scan system, and because the only surviving physical portion of the ballot (i.e. the receipt) is retained by the voter, the concern was raised by GSAÉD over keeping a strictly electronic record of the election. For example, their election procedure stipulates a date (after the election results have been ratified) by which the election ballots

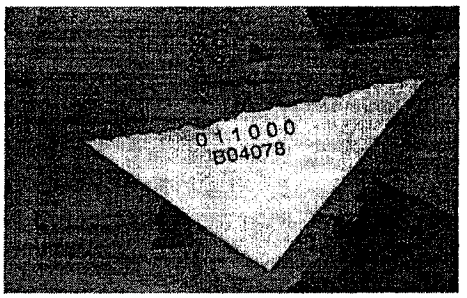

Figure 6.3: Paper-based backup "chit." Information is encoded in a succinct yet human readable way. The top row of numbers indicates the positions of the marks (0 or 1) made by the voter in each of the 6 contests. Below, the serial number is also shown and is preceeded in this case by a 'B,' indicating the voter retained the bottom page ('T' indicates a top page).

are to be shredded. Other concerns expressed by the council included power outages at the polling stations, accidental deletion, hardware/hard-disc failure, and other electronic attacks. The Punchscan team consulted with them at length about a means to provide a paper-based ballot backup.

It was agreed that a small space in the lower right-hand corner of the ballot was to be reserved for printing an encoded copy of the marked positions, which was performed at the same time as the overlays/digital signature. Using a pair of scissors, the poll worker cut off and retained this corner before returning the receipt to the voter. These paper "chits" were collected and retained by GSAÉD functioning in the stead of actual ballots. Special "craft" scissors with a patterned blade were employed. In addition to the height and angle of the cut, the special scissors add a 3rd (phase) dimension to the cut line. This was used to aid in matching a chit to its original receipt in the event of a dispute. Interestingly the paper chit was awarded "best election technology component," at the 2007 VoComp student voting systems design competition in Portland, OR.

6.1.3 Electronic Pollbook

In past GSAÉD elections, voters were authenticated at the polling place using local paper copies of the voters list. Voters' university-issued graduate student cards were used as the primary credential to vote. Although this scheme allows the eventual identification

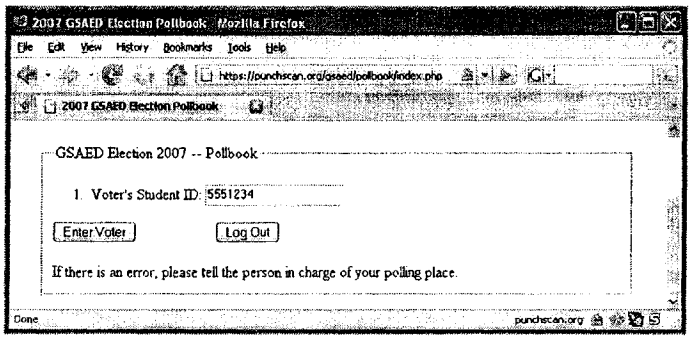

Figure 6.4: Online Electronic Pollbook used in GSAÉD Election

of students who cast more than one ballot (so-called "double voting") at different polling stations, it does not allow a means by which to invalidate the double votes from the tally. GSAÉD had experienced problems with double voting in past elections and had employed solutions such as the use of walkie-talkie radios to communicate the student numbers of voters in a real-time fashion. However the local copies of voter lists were still being updated manually which is inefficient given a list of 4000 eligible voters and 5 polling stations.

The Punchscan team offered and implemented an alternative solution: a centralized electronic pollbook. A MySQL database of eligible student numbers was created and maintained on the `punchscan.org` server. The poll worker interface was through any standard web browser and internet connection. Using the campus wireless network, the poll workers were able to log into a password protected, SSL secured web form and perform voter list queries and updates. Querying a voter's student number would return one of three possibilities: a message indicating that the student number was not found in the database, an option to mark that individual as having voted, or an alert that that student had voted already. The pollbook webpage was accessed initially by an SSL connection

6.1.4 Ballot Clipboard and Lock

Although it was agreed to be unlikely to occur in this election environment, the Punchscan team decided to test a countermeasure to the vote buying/intimidation technique known as "chain voting." If a chain voting perpetrator is able to remove one uncast ballot from a polling station without detection, they could, using the cooperation of subverted voters,

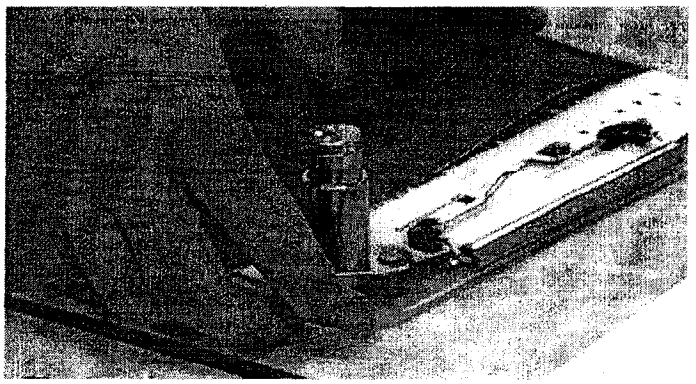

Figure 6.5: Ballot lock and clipboard used in GSAÉD election

"require that the subverted voter take the ballot to a polling place, exchange the pre-marked ballot for the blank ballot issued to that voter at the polling place, and return the blank ballot to the perpetrator to enable the next cycle" [37].

To increase the difficulty of someone being able to remove a ballot from the polling station without detection, the Punchscan team developed special clipboards that employ a *Medeco* plunger lock affixed to the clipboard. When the locking mechanism (located in the upper-right corner) was depressed, the lock's bolt would extend down into a hole in the clipboard. Each ballot also had a hole drilled in the top right corner. Before the voter is presented with an unmarked ballot it is placed on the clipboard. When the lock cylinder is depressed, the plunger would seamlessly pass through the ballot and clipboard locking it into place. When the voter returns to cast their vote, the poll worker checks to ensure that the ballot is still locked to the clipboard and that the paper corner is not torn. Additionally the clipboards were fitted with a cover to obscure from view the unique information printed on the ballot.

6.2 Preparing for the Election

There were approximately 4000 registered graduate students at the University of Ottawa, all of whom were eligible to vote. Voter turnout in past years was quite low (about 5%), and even with optimistic estimates it was not realistically expected to exceed 10% turnout for this election. Nevertheless we decided to err on the side of caution, resolving to print

3000 ballots.

6.2.1 Ballot Manufacture

Ballot manufacture refers to the process of creating holes in the appropriate locations of the physical ballot stock. However drilling holes happens to be a reasonably simple, inexpensive and available process in this application. When ballot authoring software recognizes the ballot template markers (discs and rings), it produces a special XML formated "drill file" which is used by a machinist to accurately drill holes in reams of 500 sheets of paper at a time using a computer-positioned vertical mill. As per their election procedure, the GSAÉD council met one week before the election to approve the ballot wording and layout. That evening the Punchscan team produced the drill file and took it to the machinist who drilled 3000 blank ballots. The ballots were then shipped and arrived in Ottawa the next morning.

6.2.2 The First Meeting of the Trustees

The Punchscan team implemented the threshold based secret sharing scheme previously described in this document, however given the stakes in this election were relatively low, GSAÉD exploited the option of distributing the key, and did not produce a trustee in addition to the CRO, who acted in this election as sole trustee.

At the first meeting the CRO supplied her passphrase and the open source election engine software generated all the election data from it. This includes the association of the ballot serial numbers with a pseudo-randomly generated permutation of letters. Additionally, the information needed to reconstitute, or "decrypt" the vote after one half of the ballot is destroyed is also generated. This "decryption table" will be partially revealed during the post-election audit. However to increase the probability of catching tampered ballots during the audit, more (independent) decryption tables can be created, each with a unique keystream. For this election we used 10 such tables. After this secret information is created, each piece is run through a bit commitment function. These commitments

are publically posted. Later as some of this secret information is revealed during the audits, it can be run through the same commitment function (by anyone) to ensure it matches the original commitments, establishing that the results have not been modified. These commitments, and all the election data referred to throughout this case study, are available from the Punchscan webpage[2].

6.2.3 Pre-Election Audit

After all ballots are (digitally) generated and committed to, half of the ballots are chosen at random to be audited. Because we ultimately wish to print 3000 ballots, during the election specification we (digitally) generated 6000 ballots.

The pre-election audit took place in two stages. First the CRO entered her passphrase to open and publish the information about the selected ballots. After this data had been published, the second stage was to actually audit this information against the published commitments from the first meeting. The audit report generated by the program determined that all the revealed ballots matched their commitments. It is important to note however that the audits form the core of the "independent verification" and is meant to be as available and performable by anyone interested.

6.2.4 Printing Ballots

After the completion of the pre-election audit, the CRO met with the Punchscan team to print the remaining 3000 ballots on paper. Under the supervision of the CRO, the ballots were printed in parallel on six inkjet printers. Three printers were loaned by Hewitt Packard to the University for the election (two HP-K5400's and one HP-K550), and three more were provided by the Punchscan team (HP-K550's). Although strictly speaking the printing is not dependent on a particular model of printer, a few considerations need to be taken. Firstly, the ballots contained colored scanner-alignment marks. Secondly the ballots must be accurately aligned so that the serial number and letters on the bottom

[2]http://punchscan.org/gsaed/

page show through the holes. This is usually resolved through trial and error, and using the ballot authoring software to introduce a compensation in the ballot PDF. Thirdly it was discovered for certain models that printing would always halt when the print head reach a drilled hole. Finally paper feeding was an issue because the drilling process creates a small cusp in the paper around the drill hole causing the sheets to "stick" together as they were fed into the printer. This often results in numerous pages being scrapped. However most of these issues had been previously addressed and the printing process for the 3000 election ballots took approximately one hour. Upon completion, the ballots were placed into boxes, sealed, and signed along the seal by the CRO.

6.2.5 Poll Worker Training

On the night prior to the election, the Punchscan team hosted a two-hour poll worker training program. For the first hour, an introduction to Punchscan was given explaining the theory behind the system, outlining the polling place procedures (which was also provided in written form), and answering questions. The second hour provided hands-on experience with the polling place equipment. The poll workers were provided with mock ballots to vote on and they practiced the procedure of scanning and casting the ballots.

6.3 Conducting the Election

6.3.1 Contests

The GSAÉD election consisted of six contests. Five were positions for office and one was a referendum. Of the five office positions, only one was contested. However the uncontested officials still needed to be confirmed by a majority of voters according to the GSAÉD regulations. Thus these four contests consisted of a 'yes' or 'no' option. The contested office position had two candidates, and the referendum also consisted of a 'yes' or 'no' option. In essence, the election consisted of five contests, all of which had two candidates/options.

6.3.2 Polling Station

For this election, the polling station equipment was provided by the Punchscan team to GSAÉD at no charge. There were five polling stations on each of the three days of the election. Four of the stations were distributed across the main campus, and one was hosted at a satellite campus. Each polling place consisted of a computer, printer, scanner, paper shredder, polling booth, a couple of ink daubers, and a ballot box.

Voting and Casting The ballot-issuing, marking and casting procedures were conducted in accordance with those presented in chapter 5.

6.4 After the Election

6.4.1 Election Results

After the polling stations were closed, the Punchscan team, the CRO, and a scrutineer assembled to tally the results. The encrypted ballot receipts were uploaded to the Punchscan server. The CRO then entered her passphrase to decrypt and tally the results. With 154 ballots cast, the results are summarized in Table 6.4.1.

	President M. Doumit	VP Internal K. McClellan	VP Finances R. Alamgir
Yes / Candidate A	118	120	117
No / Candidate B	28	21	31
	VP Services F. Deen-Arif	VP Communication A) P. Marchand B) T. Plante	Referendum
Yes / Candidate A	121	81	98
No / Candidate B	26	54	43

Table 6.1: GSAÉD 2007 election results

6.4.2 Online Receipt Check

At the bottom of each ballot were instructions (in English and French) directing the voter to the online receipt check website. There, voters could enter the serial number of their

ballot to verify what they were holding in their hands (i.e. the ballot receipt) is in fact what was used in the tally.

6.4.3 Post Election Audit

After the completion of the election, the tallying procedure was audited. The 10 decryption tables were each partially revealed, again depending on selections made by stock market data in similar form to the pre-election audit. The CRO entered her passphrase in order to recover the partial election data to be published online. This data was then audited with the Post-election audit software tool provided by the Punchscan team. The post-election audit verified the election data and encountered no errors or irregularities.

6.5 Incidents and Assistance Requests

In short, this election fared exceptionally well. Considering the novelty of the software and voting process, we were pleased to have not faced any incidents or assistance requests of a significant nature (i.e. any issue that was not able to be satisfactorily resolved on the spot). This section however attempts to cover the issues that were encountered by first briefly recounting the technical glitches and then to discuss the reaction of voters and poll workers to Punchscan, and more broadly to E2E elections in general.

6.5.1 Technical Issues

There were a number of issues that the pollworkers experienced. For example, of the 154 ballots cast, only 145 were recorded in the electronic pollbook. This means that nine instances occurred in which ballots were cast without the voter's eligibility being verified. However this is a matter of pollworker procedural compliance, and therefore is no different than any existing voting system in this regard.

More interesting is how the pollworkers reacted (uncued) to technological failures. On the software end, there were several instances in which either the polling place or electronic

pollbook software froze. On the hardware end, there were several instances in which the printers jammed or the wireless internet signal was lost. During these instances we found that all the pollworkers undertook to create a handwritten account[3] of the cast ballot, interestingly in some cases without having been instructed how to do so. In the absence of the printer to create overlays and digital signature, the poll workers signed the ballots manually. These handwritten records were later recorded electronically, and transcription errors would be subject to detection through the online receipt verification check. This demonstrated an unexpected robustness of Punchscan against a host of denial of service scenarios.

6.5.2 Psychological Acceptability

The act of marking a Punchscan ballot by matching shuffled letters has been referred to as "indirection". Opinions of voters varied widely over the degree to which indirection was an issue. Some voters actually claimed that Punchscan was "no harder" to mark than a traditional ballot, whereas others found it irritating. Here the voters were more concerned with their personal experience than in their ability to correctly transcribe their voting intent on a Punchscan ballot. In order to gauge the usability of Punchscan however, the rate of occurrence of intent transcription errors would have to be measured. However in the context of voting systems, a proper user-study would require the ability to maintain a linkage between individual voters and their vote in order to gauge their ability to successfully cast their vote as they had intended. Obviously this is not possible during a live secret-ballot election. However the reactions of the participants during this case-study do point to important questions a user study should undertake to answer. The design principal of psychological acceptability can be applied to voting technology [58], postulating that the security mechanisms of the system should be congruent with the voter's mental model of the system. Our case study suggests that the security mechanisms of Punchscan were not well understood by the voters.

[3]Serial number plus a numeric code for each mark (or absence of mark) made by the voter.

Ballot marking

The fact that the order of the letters on the ballots were randomized was not clearly indicated on the ballot, leaving the voter with only their intuition for forming a proper understanding of why a receipt does not contain adequate information for determining their vote. The randomized lettering was likely the predominant barrier in understanding. However once this was explained to the voters, many understood in a flash of insight. However understanding did not necessarily precipitate willingness and many voters indicated their sense of burden with the voting process. What was clear from the election was that voters did not intuitively understand the ballot marking process. Most became satisfied of the requirements after a verbal explanation. However the quality of explanations and (therefore their effectiveness) varied between the pollworkers. As of yet, there still is not an standard script on how best to educate new voters.

Shredding

Also at issue was the process of shredding one of the ballot sheets. Concern was expressed by several voters over what they perceived as the destruction of their vote. Many were also confused by being given the option to shred either page, in some cases asking several times to ensure they understood correctly. This might be attributed to the fact that typical administrative tasks (such as filling out a government form) do not offer options as to how the task should be completed. Therefore we might reasonably conclude the choice component of ballot completion is contrary to the voter's mental model. The original design intention was for voters to retain top and bottom sheets with near equal probabilities to reveal ballot tampering. Interestingly however given the choice approximately 85% of the voters chose to retain the bottom page as their receipt. On the third day of polling, after this trend had been established, the Punchscan team questioned voters leaving the polling station why they chose the sheet to shred that they did. The majority explained that because the ballot was still locked to the clipboard, they found that ripping off the top sheet to be the easiest way to complete the action. However in a subsequent Punchscan

election, held at the 2007 Computers, Freedom and Privacy (CFP) conference, where clipboards were not used, approximatly 85% of the 36 voters still chose to keep the *top* sheet instead. The implication of receipt skew is that it is less probable that attacks will be detected during the post-election audit. Although the degree of receipt skew in this election didn't significantly affect a reasonable assurance of integrity, it is something that should be better understood for future elections.

Conveying the Purpose of Punchscan

Another source of dissonance between the ideals of Punchscan and voters' mental model concerned the purpose of ballot receipts. Many did not realize they were going to receive a receipt and wanted to leave immediately after marking their ballots. Furthermore, when it was explained to them that they would receive a receipt, and what the receipt allowed them to do with respect to independent verification, a number of voters were still indifferent. It is important to the integrity of the election that voters at least take the receipt—a left receipt means that receipt will not be checked, opening a window of opportunity for an attacker to modify that ballot.

Voters' reactions demonstrated that E2E has a long way to go in making its benefits clear to voters. Its not necessary for the voters to have a perfect mental model of the system. It is difficult to gauge how fundamental these problems are to Punchscan, and E2E systems in general. They could be simply rooted in the novelty of this kind of system, in which case voter education would go a long way to resolving these issues. Our case study does indicate that as some voters became aware of the verification ability afforded to them with Punchscan, they were accepting of the extra measures required for voting.

The ballot receipts were hosted on the Punchscan webserver and the server logs show that the image files of 83 of the 154 ballots got at least one hit. Whether this means they were actually checked against the receipt cannot be determined from available data, but it is suggestive of an interest by voters in checking their ballot receipts online.

Poll Worker Feedback

The poll workers' opinions varied as to the difficulty of the ballot casting procedure. All polling stations encountered minor computer glitches at some point during the election, as well as the occasional printer jam. The polling place software also experienced some buggy behavior after being left running for long periods. However recalling from earlier, on the occasions that there were technical complications, the poll workers were able to record the ballots manually. Their comments include the need for "a more detailed explanation of the voting process on the ballots." One poll worker explained to us that a voter daubed the serial number holes, which in turn caused a serial number recognition error at the polling station. Also echoing the comments of voters was the need for an explanation as to "why the voters have to destroy a page." They also suggested visual instructions posted inside the polling booth, a brochure explaining Punchscan's "security features."

Perhaps the most widespread concern however was the time to cast. Obviously 154 voters across 5 stations and 3 days allowed for the poll workers to take their time casting votes. However, as one poll worker described it, "I don't know how it will work if we have people lining up." We did not conduct stopwatch-timed trials during the actual election, however generally we found unlocking the clip board, scanning and casting the ballot, and then printing overlays took around 60 to 90 seconds. On the other hand, at the subsequent CFP election we used Punchscan in what we call "mail-in" mode, which in this context meant that instead of shredding one half of the receipt, the voter kept one half, and gave the other half to the poll worker. The collected receipts are scanned at a later date. Therefore the time to cast for this variant was the time it took to hand the poll worker one of the ballot sheets (effectively zero). However the mail-in version does not offer the same degree of privacy or integrity as the full-scale version used in the GSAÉD election.

Version	Time to Cast	Integrity	Privacy
Clipboard, Scan, and Overlays	60-90 seconds	Very High	Very High
Scan and Overlays	40-70 seconds	High	High
Scan Only	20-40 seconds	Medium	High
Mail-in	~	Medium	Medium
Standard Paper Ballot	~	Low	High

Figure 6.6: Time to vote statistics. These statistics do not include time to mark a ballot, only the time added by the Punchscan architecture over-and-above a standard paper ballot.

6.6 Metrics

6.6.1 Average time to vote

The average time to vote varies according to the additional features overlays and receipts used. Figure 6.6 summarizes this information. We did not conduct thorough time tests during the election, and these statistics are approximations.

6.6.2 Cost

The cost of a Punchscan election can be considered in terms of fixed and marginal costs. The fixed cost is the cost to cast the first ballot, and it includes all the overhead costs. The marginal cost is the cost to cast an additional ballot given that the fixed costs have been accounted for. We now consider these costs per polling place in Canadian dollars.

We estimate the cost of the equipment as follows: a basic computer capable of running JRE ($200), a flatbed scanner ($100), an ink jet printer ($200), a paper shredder ($50), a clipboard and lock ($20), an ink dauber ($1), a polling booth ($10), and ballot box for the paper backup receipts ($10).

The ballots must be drilled, printed, and then have overlays printed over them. At 19 holes per ballot and 6 reams of paper, the setup and drilling was completed in under half an hour. Given the machine and operator rate of $75/hour, 3000 Punchscan ballots were produced at a unit cost of $0.01. Using standard quality white 8.5x11" office paper costing

$0.01 per sheet, the overall until cost of an unprinted Punchscan ballot was $0.03. Hewitt Packard lists the cost per page (CPP) of printing black text on the HP K550 Officejet Pro as $0.015. A Punchscan ballot is essentially black ink, with only small yellow alignment marks. The overlays are color however they are sparse and not near a full page of text. We estimate the CPP of printing the ballot and the overlays to be $0.03, bringing the unit cost of a ballot to $0.06.

Poll workers were paid $10 per hour and two poll workers were at each station. Average time to vote is 1.5 minutes, so wages for a vote are $0.51.

Totaling these costs together, the fixed cost to cast the first ballot is $591.57. The marginal cost to cast an additional ballot is $0.06 (or $0.57 with labour).

6.7 System Performance

In the end, this election was successful in the sense that its participants (the Punchscan team, GSAÉD, the voters) were sufficiently satisfied in Punchscan's ability to fairly and accurately conduct this election. The elected candidates became ratified by the council, and no challenges were made against the election results. The election was modest and not hotly contested, but it is our position that this case study represents an important milestone for Punchscan and E2E elections in general. By all accounts the 2007 GSAÉD election was a success.

Chapter 7

Discussion and Conclusion

7.1 Summary of Security Design

In this section we return to the integrity and privacy criteria presented in chapter 2 and briefly summarize how Punchscan has approached them in the context of the voting problem.

7.1.1 Integrity Protection Measures in Punchscan

Marked as intended Punchscan, and indeed no paper ballot and/or optical scan system, provides direct protection against marked as intended errors. The assumption is that the voter is able to verify the correctness of their own marking at voting time (and for ballot secrecy reasons, voting time only). DRE touch-screen voting machines do provide a measure of protection; they often provide feedback of the voter's choices and ask for the voter's confirmation of the choices before the vote is recorded. However such machines suffer greatly in the other two spheres of integrity. Ultimately marked as intended errors are minimized through voter education and usable design–two vital areas most often neglected by security researchers.

Recorded as marked After voting, the voter retains a portion of their paper ballot as a receipt (which includes the original mark made by the voter). In order to protect privacy

the mark does not demonstrate how the voter voted, but because the mark is original, the voter knows it reflected their voting intent. Also, the receipts contain non-private information, meaning they can be collected and published in a newspaper, or an online database. We chose to implement an online receipt checking tool that allows the voter to enter a serial number on a website and see the officially recorded version of their mark. A paper receipt that matches the official account constitutes proof that the mark recorded by the voting machine was the mark made by the voter. Additionally, assuming the receipt is signed by the election administration, it can potentially validate a claim of error/fraud when the marks on the receipt and official record do not match.

Counted as recorded When the election is concluded and the official results are released by the election administration, a public audit period is held. Certain portions of information held by the election administration are "challenged" to be publically revealed. A series of cryptographic checks are performed using free open source software tools to ensure self-consistancy and therefore correctness of the counting. In practice an additional "printing" audit is held before the election to ensure the ballots cannot be improperly constructed to allow a vote to be counted for anyone else than the correct candidate. How to design this information to prove the counted as recorded property while maintaining privacy represents a major cryptographic design challenge, and is one of the primary contributions of Punchscan.

7.1.2 Privacy Protection Measures in Punchscan

The aforementioned examples that carry the potential to violate voluntary and involuntary privacy have been addressed in the design of Punchscan.

Voluntary Privacy With respect to voluntary privacy, the receipts are "encrypted." We've shown in [16] that the Punchscan receipts provide "no non-negligible information," meaning in the absence of the decryption keys, provides no useful information about how the voter voted.

Involuntary Privacy The use of these ballot receipts is also employed in the protection to involuntary privacy. Because the pollworkers only record the ballot receipt, the data stored in electronic memory is encrypted. Not only does this dispense with the need for securely random memory mappings, but coupled with the integrity features, it also dispenses with the need for a trusted software platform at the polling place. This carries serious implications for voting machine certification in the United States. In mid 2007, New York state proposed an amendment to their election law to require a source code review of all election system technology as a condition of certification [45]. The law requires vendors to place their source code into escrow for review. This would potentially limit the use of closed platforms by vendors such as Microsoft. The privacy and integrity assurances of Punchscan are software independent[1] with respect to the polling place software implementation. This may provide a justification for the relaxation of source code review requirements of polling place equipment that may potentially allow closed platforms.

7.2 Punchscan in the scope of E2E

Punchscan has been designed to be an E2E voting system. Most of the system components presented in the previous chapters were developed to realize the E2E criteria. The EAC found that the range of proposed E2E systems have points of commonality, and they attempted to summarize these in a list of properties. Here we present some of them and briefly explain how Punchscan does or does not exhibit their properties. Note that this is not a list of requirements for a system to be classified as E2E but rather a preliminary sketch of the typical properties of these systems.

Property 1. *Voters' ballot selections are encrypted for later counting by designated trustees.*

A Punchscan ballot consists of two pages. The top page contains a list of contests and can-

[1]Software independence as defined in the 2007 draft VVSG requires that "an undetected error or fault in the voting system's software is not capable of causing an undetectable change in election results," [23].

didates with set of randomly ordered symbols beside the candidate names. There are holes in the top sheet that display a corresponding (but independently and randomly ordered) set of symbols. To vote on a Punchscan ballot, the voter observes the symbol appearing beside their chosen candidate's name, and locates the matching symbol in the holes. The voter then marks that hole with an implement such as a bingo-style dauber. The implement is sized slightly larger than the hole such that the ink mark will be made on both sheets. One of these sheets is destroyed in a cross-cut paper shredder. The remaining sheet represents the voter's receipt and is now "encrypted.[2]" Only the threshold number of trustees (aka the election authority) have the ability to reconstruct the information contained on the destroyed sheet.

Property 2. *Voting will produce a receipt that would enable the voter to verify that their ballot selections were recorded correctly and counted in the election.*

Punchscan uses a robust audit procedure, including a process by which a voter can visit the election website and look up their ballot using the serial number contained on their receipt and verify what they hold in their hand matches what was recorded by election authority.

Property 3. *The receipt preserves voter privacy by not containing any information that can be used to show the voter's selections.*

Because one of the sheets is shredded, and assuming that the ordering of symbols contained on that page were uniformly random and independent from the page that was retained (aka the receipt), then no information about the destroyed sheet is contained on the retained sheet, and therefore the vote cannot be guessed with any advantage.

Property 4. *No one designated trustee is able to decrypt the records; decryption of the records is performed by a process that involves multiple designated trustees.*

[2]Since both sheets contain random but independent orderings of the symbols, possessing only one of the sheets does not give you information about the corresponding symbol on the other sheet.

Punchscan employs a threshold based password scheme whereby a pre-designated number of trustees must correctly enter their passwords before the records can be reconstructed.

Property 5. *End to end systems store backup records of voter ballot selections that can be used in contingencies such as damage or loss of its counted records.*

Punchscan in its original form relies on voter receipts to reconstruct an election should the counted records be destroyed. However, as will be discussed in the next section, the implementation of Punchscan used in our case study expanded the originally proposed system to include a paper-based backup of the ballot receipts.

Property 6. *The backup records contain unique identifiers that correspond to unique identifiers in its counted records, and the backup records are digitally signed so that they can be verified for their authenticity and integrity in audits.*

The backup ballot receipts used in our election contained a serial number which matched the serial number of the ballot. While the ballot receipts themselves were digitally signed, the paper backups were not as it was agreed that the backup records were very unlikely to be needed. Should they have been used, they would have been published and thus their integrity would be ensured through voter verification. In future elections, consideration will be given to digitally signing the backups as well as the receipts.

Property 7. *The documentation includes extensive discussion of how cryptographic keys are to be generated, distributed, managed, used, certified, and destroyed.*

The source code for all the software used by Punchscan is open source and can be examined by anyone. Furthermore, the Punchscan team has attempted to document the underlying cryptography of the system through papers, presentations, and other documentation available from the Punchscan website[3].

Property 8. *Vote capture stations used in end to end systems must meet all the security,*

[3]http://punchscan.org

usability, and accessibility requirements.

The security of the vote capture station in a Punchscan election is similar to that of a paper ballot voting station. Two additional security measures are taken: one is to lock the ballot to a clipboard and the second is to ensure a high-integrity paper shredder. The purpose of this case study, in part, is to examine the usability of Punchscan and will be discussed further in later sections.

Property 9. *Reliability, usability, and accessibility requirements for printers in other voting systems apply as well to receipt printers used in end to end systems.*

Punchscan can be easily implemented with inexpensive off-the-shelf equipment. As will be examined further in this case study, reliability and usability issues emerged. However Punchscan is largely hardware independent and could be adapted to use proprietary voting-dedicated equipment.

Property 10. *Systems for verifying that voter ballot selections were recorded properly and counted are implemented in a robust secure manner.*

Punchscan allows the voter to verify the proper scanning of their ballot at the polling station before it is cast, in addition to their ability to check the receipt online. The security of the Punchscan tallying process is dependent on well-studied cryptographic primitives [51] and no implementation vulnerabilities have been discovered to date.

7.3 Design Comparison

Consider the key-distribution protocol Kerberos (a sufficient description can be found in 12.14 of [43]). It attempts to solve the key distribution problem: how can two entities, whom have no prior relationship, securely establish a shared secret-key? The Kerberos protocol uses a symmetric encryption based approach with the help of a trusted sever. However at the end of the day, each entity engaging in the protocol must have a pre-shared secret key with the trusted server. How are these initial keys distributed? We can see that

Kerberos is able to reduce $O(n^2)$ key distributions to $O(n)$ key distributions–that is to say, it does not solve the key-distribution problem entirely.

Punchscan is, in some respects, like Kerberos. It is attempting to solve what we'll refer to as the "DRE problem:" unverifiable recording and counting of votes in an election. But it doesn't solve the problem fully. The design of the Punchboard has allowed us to eliminate the DRE problem at all polling locations, as well as satisfy the integrity criteria of election results. However the generation of the Punchboard as well as the printing of the ballots still offers DRE style privacy assurance. We've attempted to mitigate this through the open-source verifiable election engine software. However ultimately its execution places trust in the hardware of the diskless-workstation. In that sense you could say, with respect to privacy, Punchscan is a DRE and inherits the DRE problem. However our claim is that because the number of DRE's are reduced from many (across all polling locations) to one (across the entire election), our ability to operate this DRE in a safe and controlled environment is much higher than any current DRE-based election.

Therefore we can reasonably claim that Punchscan improves tremendously on the integrity and privacy factors of the DRE problem. The DRE-VVPAT combination improves on integrity assurance also, but does not necessarily improve privacy, meaning that Punchscan offers a privacy advantage over it as well. However we assert that Punchscan also improves on the DRE-VVPAT integrity assurance. Assuming that a VVPAT implementation functioned perfectly as intended, the counted as recorded integrity criterion could still only be proven if the verifier,

1. Participates in chain-of-custody of the VVPAT record from voting time to verification time,

2. Manually audits (i.e., tabulates votes of) the entire VVPAT.

In practice this is infeasible. Conversely because of Punchscan's use of cryptographic commitments, *the informational equivalent of chain-of-custody is established when the verifier downloads the commitments before the election.* Because the audit process is automated, the verifier can audit the entire election in a few mouse clicks. Because a Punchscan

audit is vastly simpler to perform than the equivalent manual VVPAT audit, we reasonably expect more people to participate in the verification process. But more importantly, it allows the process to be verified end-to-end. VVPAT being a physical verification medium requires physical supervision. Because its not realistic to be physically present across the entire functional lifetime of the VVPAT, we cannot make the same claim of end-to-end verification of it.

How does our approach compare to other E2E systems? As was mentioned in chapter 2, Three Ballot requires a DRE-equivalent to validate ballots meaning it inherits the DRE problem. Even neglecting this aspect, there are several privacy/integrity attacks, including those we developed in [16]. A more interesting comparison comes between Punchscan and Prêt à Voter (PAV). These systems are remarkably similar in their feature set, but there are still differences. PAV also has ballot custody privacy problems, but having only one sheet(i.e., one permutation) cannot address the problem with independently printed ballot sheets as Punchscan does. Perhaps most notably, the PAV ballot is simpler to mark than in Punchscan. How much simpler, and how that may translate into voting accuracy have yet to be meaningfully tested. PAV also does not require a diskless workstation to decrypt the ballots but rather a MIX net. This is arguably better for privacy because there's no trusted hardware required and the DRE problem is, in this way, avoided. However there remains an authority knowledge problem surrounding the *generation* of the ballots. Some preliminary solutions have been suggested, but not implemented and deployed as is the case of the Punchscan diskless workstation.

The problems inherent to the original incarnation of PAV, (PAV 2005) [11] were numerous. Each ballot required some 2^{40} bits of randomness as well as 2^{12} *public key* and 2^{11} symmetric encryption/decryption operations to construct the onion. Conversely Punchscan requires 2^4 permutation (i.e., memory lookup) operations generated by some 2^7 *symmetric* encryption operations[4]. Roughly speaking, PAV 2005 requires about 2-3 orders of magnitude more computations to perform encryption/decryption meaning it would take hours if not days to compute election results that an equivalently sized Punchscan election could

[4]Assuming an 8-candidate election with 10 instances of the d-table)

compute in minutes. Generation and verification of the audit data is also much more computationally heavy, though precise comparisons have not yet been made. There were many improvements to PAV 2006 [57], including the ability for a voter to conduct what would amount to the equivalent of the Punchscan pre-election audit. It is mentioned in [66] that a second onion is used "which is encrypted under the public key of the voting machine, is decrypted when voters cast their votes," suggesting that the polling machine would be responsible for secret key management and trusted decryption–similar to the very DRE model we're attempting a departure from.

In the end the success of one system over another will rest in its practicality–for Punchscan, the ballot; for the Pret a Voter, the system performance. The commonalities of the system have lead to suggestions of merging aspects of the two. A "best of both worlds" scenario may be the PAV style ballot encrypted under the Punchboard architecture.

7.4 Future Work

The work presented in this thesis represents a solid beginning at end-to-end verification of electronic vote-counting systems. The ultimate goal of system design however is to produce a result that will eventually see use. Although the system we've developed offers unprecedented security of election results, there are many other factors to design that need be addressed.

7.4.1 Usability Testing

Punchscan was roundly criticized at VoComp for its usability. The criticism has been mainly informal and though to date no formal user study has been conducted to confirm these criticisms, the perception is sufficiently widespread to demand formal study in this area. There are two key issues at stake:

- Voter perception and experience,
- Voter accuracy.

Much of the discussion has centered around the former–how the voters "feel" about marking a Punchscan ballot. However it is our contention that the more important measure, from a democratic standpoint, is how accurate voters are at transcribing their intent. As was postulated in [7], the higher conceptual complexity of the Punchscan ballot may force voters to give more conscious thought to the marking process, and improve accuracy in so doing. Specifically we propose that future work seek to establish and compare intent transcription accuracy rates for a range of systems including Punchscan, Pret a Voter, 3-ballot and traditional optical scan ballots through user tests.

Although the highest profile in the context of usability, recall that voters are only one of four groups involved in a Punchscan election. The procedures outlined in chapter 5, when compared to those of a conventional election, would reveal extra steps that have to date escaped realistic evaluation of usability. We need to understand more about how (or if) independent verification burdens the trustees and poll workers. Additionally we need to gain a sense about the independent verification itself; the more difficult to perform, the less it will be performed. Finally it would be of value to pursue more general usability questions as well: can ballots or other components of the system be easily customized to suit the conditions of a particular election environment? The research outlined here will be a priority for establishing Punchscan and E2E systems as viable solutions.

7.4.2 Formal Design Criteria and Standards Compliance

Initially the 2005 draft VVSG helped guide E2E research. With the removal of the E2E from the 2007 draft, the momentum of E2E research has shifted. Indeed the notion of an "innovation class" for our purposes is paradoxical: our research is needed to shape the innovation class which is needed to shape our research.

As it stands, the term "E2E" itself is not even present in current EAC discussion surrounding the Innovation class, meaning that the EAC and researchers are, at present, not speaking the same language. The E2E criteria listed above were also removed. The requirements for system certification under the innovation class are broad, and subject to

interpretation. For example, the notion of "independent verification," (IV) has been replaced with the term "software independence," (SI). It appears to have been the intent to position SI as a more general requirement, however the opinion here is that the opposite is true–it more narrowly and inconsistently captures the notion of independent bodies being able to verify election results.

What is required now is a consensus on terminology and key concepts to be able to move forward. Does the research community direct its energy to establishing common criteria for the innovation class before serious design can proceed, or do we let our continued work into system design dictate the eventual criteria? Certainly this is the most relevant and immediate issue to the Punchscan project. Time is short–the comment period for the 2007 draft ends in mid 2008, barely after the publish date of this thesis. Once these comments are incorporated into the final VVSG draft by the EAC, the focus of research will take on a more decisive direction; conforming Punchscan or its decendants for eventual certification under the innovation class.

7.5 Conclusion

Though this thesis has drawn to an end, the development of practical E2E systems has really only begun. The need for secure and verifiable elections will only increase as our reliance on electronic equipment in elections increases. Efforts like system certification, source code review and physical tampering countermeasures can only take the integrity assurance half way to being satisfactory. The first attempt at full integrity assurance of electronic systems–VVPAT–has been enlightening, but is ultimately not a rational solution. Electronic vote-counting systems were developed for the purpose of automating the tediously slow manually counted paper-based system. However addressing the observability and verifiability concerns these electronic systems introduce by implementing a parallel manually counted paper-based system is a circular solution–designing a new system to improve upon the original system but introducing new problems that are solved by using the original system. This is partially because the security of optical scan/DRE systems was

an afterthought. One of the tenets of secure systems design is that security be designed into the system from the *beginning*, not added on afterward.

In this thesis we've presented the design of an optical-scan vote-counting system in which security was designed *into* the system. In chapter 2 we presented a set of integrity and privacy criteria to guide the design, and introduced the notions of independent verification and end-to-end (E2E) verification and gave examples of such systems. In chapter 3 we began discussing Punchscan with a high level architectural sketch of how these design criteria would be realized. Then we moved to a more detailed discussion in chapter 4 of the functionality of the implementation with a complete list of independent verification algorithms. Chapter 5 presented a time line and set of procedures for a Punchscan election. The concepts presented in the first five chapters were applied in chapter 6 to a case study of Punchscan in a live election. Finally we've concluded here in this chapter with a discussion of how Punchscan met its design goals, which in turn has pointed us to future directions of research.

Although Punchscan is not the final word in E2E design, it is, as presented in this thesis, the first to be developed to the point to where it has seen real-world use. The real challenge comes now–developing an E2E system to the point where it will see widespread use.

Bibliography

[1] B. Adida and R. Rivest. Scratch & Vote: self-contained paper-based cryptographic voting. Proceedings of the 5th ACM workshop on Privacy in Electronic Society, pages 29-40, 2006.

[2] R. M. Alvarez, M. Llewellyn, T. E. Hall. Are Americans Confident Their Ballots Are Counted? *CalTech/MIT Voting Technology Project*, paper #49, 2006. `www.vote.caltech.edu/media/documents/wps/vtp\_wp49.pdf`

[3] L. Atkeson, K. Saunders. The Effect of Election Administration on Voter Confidence: A Local Matter? PS: Political Science & Politics conference, *American Political Science Association*, 40.4, 2007.

[4] Matt Bishop. Overview of Red Team Reports, *University of California*, 2007. `http://www.sos.ca.gov/elections/voting\_systems/ttbr/red\_overview.pdf`

[5] Brennan Center Task Force on Voting System Security. The Machinery of Democracy: Protecting Elections in an Electronic World. *Brennan Center For Justice*, 2006. `http://www.brennancenter.org/dynamic/subpages/download_file_39288.pdf`

[6] The Machinery of Democracy: Voting System Security, Accessibility, Usability and Cost. *Brennan Center For Justice*, 2006. `http://www.brennancenter.org/dynamic/subpages/download\_file\_38150.pdf`

[7] J. Buechler, T. Earnet, and B. Smith. Voting System Usability: Optical Scan, Zoomable, Punchscan. *UMBC CMSC 691/491V – Electronic Voting by Alan T. Sherman*, May 2007.

[8] J. Calandrino et. al. Source Code Review of the Diebold Voting System. *University of California, Berkeley*, 2007. http://www.sos.ca.gov/elections/voting_systems/ttbr/diebold-source-public-jul29.pdf

[9] Decertification/Recertification Decisions of Voting Machines. *California Secretary of State's Office*, 2007. http://www.sos.ca.gov/elections/elections_vsr.htm

[10] R. T. Carback, S. Popoveniuc, A. T. Sherman, and D. Chaum. Punchscan with Independent Ballot Sheets: Simplifying Ballot Printing and Distribution with Independently Selected Ballot Halves. Proceedings of *Workshop on Trustworthy Elections 2007.*

[11] D. Chaum, P. Ryan, and S. Schneider. A practical voter-verifiable election scheme. Technical Report of University of Newcastle, CS-TR:880, 2005.

[12] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2): 84-88 (1981).

[13] D. Chaum. Secret-Ballot Receipts: True Voter-Verifiable Elections. *IEEE Security and Privacy*, IEEE Computer Society, 02.1, Los Alamitos, CA, USA, 2004, 38–47.

[14] D. Chaum, C. Crepeau, and I. Damgard. Multiparty unconditionally secure protocols. Proceedings of the twentieth annual *ACM Symposium on Theory of Computing*, 1988.

[15] J. A. Cheibub, et al. Electoral System Design: the New International IDEA Handbook. *International Institute for Democracy and Electoral Assistance*, 2005.

[16] J. Clark, A. Essex, and C. Adams. On the security of ballot receipts in E2E voting systems. Proceedings of *Workshop on Trustworthy Elections 2007.*

[17] J. Clark, A. Essex, and C. Adams. Secure and observable auditing of electronic voting systems using stock indices. *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) 2007.*

[18] A. Cordero, D. Wagner, and D. Dill. The Role of Dice in Election Audits. *IAVoSS Workshop On Trustworthy Elections*, 2006.

[19] J. Daemen and V. Rijmen. The Design of Rijndael. *Springer-Verlag*, 2002.

[20] D. Dill. VVPR Attack with Misprinted VVPAT. Threats to Voting Systems, *National Institute of Standards and Technology*, 2003.

[21] W. Edelstein. New york state law and lever voting machines. *New yorkers for verified voting*, 2006. `http://www.nyvv.org/doc/NYSLeverMachines.pdf`

[22] 2005 Draft Voluntary Voting System Guidelines, *Election Assistance Commission*, Version 1.0, 2005. `http://www.eac.gov/voting\%20systems/docs/vvsgvolume\_ii.pdf/attachment\_download/file`

[23] 2007 Draft Voluntary Voting System Guidelines, *Election Assistance Commission*, 2007. `http://www.eac.gov/vvsg`

[24] 2004 Voting Equipment Study. *Election Data Services, Inc.*, 2004. `http://www.edssurvey.com/images/File/VotingEquipStudies\%20/ve2004\_report.pdf`

[25] 2006 Voting Equipment Study. *Election Data Services, Inc.*, 2006. `http://www.edssurvey.com/images/File/VotingEquipStudies\%20/ve2006\_report.pdf`

[26] Poll Clerk and Deputy Returning Officer Training Guide. *Elections Ontario*, 2007.

[27] A. Essex, J. Clark, R. Carback, and S. Popoveniuc. Punchscan in practice: an E2E election case study. Proceedings of *Workshop on Trustworthy Elections 2007.*

[28] E. A. Fischer. Voting Technologies in the United States: Overview and Issues for Congress. *Congressional Research Service (CRS) Report for Congress*, 2001.

[29] K. Fisher. Punchscan: Introduction and System Definition of a High-Integrity Election System. Masters thesis, *University of Maryland-Baltimore County*, 2006.

[30] K. Fisher, R. Carback and A.T. Sherman. Punchscan: introduction and system definition of a high-integrity election system. Proceedings of *Workshop on Trustworthy Elections 2006.*

[31] J. Gill. Computational complexity of probabilistic Turing machines. SIAM Journal on Computing, 6 (4), pp. 675-695, 1977.

[32] Help America Vote Act, *United States of America Pub.L.* 107-252, 2002.

[33] B. Hosp and P. L. Vora. An Information-Theoretic Model of Voting Systems. Threat Analyses for Voting System Categories: A Workshop on Rating Voting Methods, 2006.

[34] H. Hursti. Critical Security Issues with Diebold Optical Scan Design. *Black Box Voting, Inc.* 2005.

[35] M. Jakobsson, A. Juels, and R. L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. Proceedings of the 11th USENIX Security Symposium, pages 339-353, 2002.

[36] D.W. Jones. A brief illustrated history of voting. University of Iowa, 2003. `http://www.cs.uiowa.edu/~jones/voting/pictures/\#lever`

[37] D.W. Jones. Chain voting. *Workshop on Developing an Analysis of Threats to Voting Systems*, National Institute of Standards and Technology, 2005.

[38] D.W. Jones. Punched Cards–A brief illustrated technical history. University of Iowa, 2006. `http://www.cs.uiowa.edu/~jones/cards/history.html`

[39] A. Kent. Unconditionally secure bit commitment. *Physical review letters*, American Physical Society, 83.7, 1999.

[40] A. W. Lo and A. C. MacKinlay. *A Non-Random Walk down Wall Street.* Princeton University Press, 1999.

[41] B. Malkiel. *A Random Walk Down Wall Street.* W. W. Norton & Company, 1973.

[42] B. B. Mandelbrot, Richard L. Hudson. *The (mis) Behaviour of Markets: A Fractal View of Risk, Ruin and Reward.* Basic Books, New York, 2004.

[43] A. Menezes, P. van Oorschot, S. Vanstone. Handbook of applied cryptography. CRC Press, Boca Raton, 1997.

[44] R. Mercuri. Physical Verifiability of Computer Systems. PhD dissertation, *University of Pennsylvania*, 1992.

[45] Voting machine or system; requirements. *Laws of New York* ch. 181, sec. 7-208, 2005. `http://www.elections.state.ny.us/NYSBOE/hava/Chapter\%20181\%20Voting\%20Machines.pdf`

[46] FIPS 180-2: Secure Hash Standard. *National Institute of Standards and Technology*, 2002.

[47] B. O'Neal. Electoral Systems. *Political and Social Affairs, Parliament of Canada,* BP-334E, 1993.

[48] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. *17th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 223-238, 1999..

[49] J. Pammett, L. LeDuc. Explaining the Turnout Decline in Canadian Federal Elections: A New Survey of Non-voters. *Elections Canada*, 2003.

[50] Democrats Hold Enthusiasm, Engagement Advantage November Turnout May Be High. *Pew Research Center.* 2006.

[51] S. Popoveniuc and B. Hosp. An introduction to Punchscan. Proceedings of *Workshop on Trustworthy Elections 2006.*

[52] S Popoveniuc and J. Stanton. Undervote and Patter Voting: Vulnerability and a mitigation technique. Proceedings of *Workshop on Trustworthy Elections 2007.*

[53] S. Reiss. The Wired 40. *Wired*, 14.07, July 2006.

[54] R. Rivest. Three Voting Protocols: ThreeBallot, VAV, and Twin. Invited lecture, *University Student Voting Competition (VoComp)*, Portland, OR., 2007.

[55] R. Rivest, W. Smith. Three Voting Protocols: ThreeBallot, VAV, and Twin. USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07), 2007.

[56] Ronald Rivest. Personal communication. September 18th, 2007.

[57] P. Ryan and S. Schneider. Pret a Voter with re-encryption mixes. Technical Report of University of Newcastle, CS-TR:956, 2006.

[58] J. Saltzer and M. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63:9, 1975.

[59] R. Saltman. Effective Use of Computing Technology in Vote-Tallying. *Clearinghouse on Election Administration*, 1975. `http://csrc.nist.gov/publications/nistpubs/NBS_SP_500-30.pdf`

[60] F. Schaffer, A. Schedler. What Is Vote Buying? The Limits of the Market Model. *Poverty, Democracy, and Clientelism: The Political Economy of Vote Buying*, Stanford University, 2005.

[61] M. Shamos. Paper Trail Manipulation. Threats to Voting Systems, *National Institute of Standards and Technology*, 2005.

[62] Voter Confidence and Increased Accessibility Act. *United States of America Bill H.R. 811*, 2007. `http://www.govtrack.us/congress/bill.xpd?bill=h110-811`

[63] J. Whack. Threat to voter privacy with voter verified paper audit trail voting systems using spooled paper rolls. Threats to Voting Systems, *National Institute of Standards and Technology*, 2005.

[64] John Whack. Personal communication. September 18th, 2007.

[65] D. Wu. Regression Analyses on the Butterfly Ballot Effect: A Statistical Perspective of the US 2000 Election. *International Journal of Mathematical Education in Science and Technology*, v33 n2 p309-317 Mar 2002.

[66] Z. Xia, S. Schneider, J. Heather, P. Ryan, D. Lundin, R. Peel and P. Howard. Pret a Voter: All-In-One. Proceedings of *Workshop on Trustworthy Elections 2007.*

Appendix A

Glossary of Acronyms

AES: Advanced Encryption Standard

CRO: Chief returning officer

DRE: Direct recording electronic vote-counting machine

EAC: American Election Assistance Commission

E2E: End-to-end verification

GSAÉD: University of Ottawa's Graduate Students' Association / Association des étudiant(e)s diplômé(e)s

IV: Independent verification

NIST: American National Institute of Standards and Technology

PAV: Prêt-à-Voter

VVPAT: Voter-verifiable paper audit trail

VVSG: Voluntary Voting Systems Guideline