

Security Matters: Privacy in Voting and Fairness in Digital Exchange

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op dinsdag 25 augustus 2009 om 16.00 uur

door

Hugo Lennaert Jonker

geboren te Eindhoven

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. S. Mauw
en
prof.dr. J.C.M. Baeten

Copromotor:
dr. J. Pang

A catalogue record is available from the Eindhoven University of Technology Library
ISBN: 978-90-386-1906-4



Ph.D.-FSTC-10-2009
The Faculty of Sciences, Technology and Communication

DISSERTATION
Defense held on 25/08/2009 in Eindhoven
in candidature for the degree of

**DOCTOR OF THE UNIVERSITY OF
LUXEMBOURG**
IN COMPUTER SCIENCE

by

Hugo JONKER

Born on 9 August 1978 in Eindhoven

**SECURITY MATTERS: PRIVACY IN VOTING AND FAIRNESS IN
DIGITAL EXCHANGE**

Dissertation defense committee

Dr. S. Mauw, first promotor

Professor; Université du Luxembourg

Dr. J.C.M. Baeten, chairman and second promotor

Professor; Eindhoven University of Technology

Dr. P.Y.A. Ryan

Professor; Université du Luxembourg

Dr. S. Etalle

Professor; Eindhoven University of Technology

Dr. B.P.F. Jacobs

Professor; Eindhoven University of Technology

Dr. J. Pang, copromotor

Université du Luxembourg

Promotoren:

prof.dr. S. Mauw (University of Luxembourg)

prof.dr. J.C.M. Baeten (Eindhoven University of Technology)

Copromotor:

dr. J. Pang (University of Luxembourg)

Kerncommissie:

prof.dr. P.Y.A. Ryan (University of Luxembourg)

prof.dr. S. Etalle (Eindhoven University of Technology)

prof.dr. B.P.F. Jacobs (Eindhoven University of Technology)

© 2009 H.L. Jonker

IPA Dissertation Series 2009-15

University of Luxembourg Dissertation Series PhD-FSTC-10-2009

Printed by University Press Facilities, Eindhoven

Cover design by R.B. Jonker



The work on this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics). The author was employed at the Eindhoven University of Technology in the BRICKS PDC1.3 project “Protocols for secure Infrastructure and E-commerce”, and he was employed at the University of Luxembourg in the National Research Fund project BFR07/030 “A Formal Approach to Privacy in Electronic Voting”.

Acknowledgments

In writing this thesis, I enjoyed the support of many. First of all, I am grateful to the members of the core committee, Peter Ryan, Bart Jacobs, and Sandro Etalle, for the time and effort they spend in reviewing this thesis and sharing their findings with me. I would also like to thank my direct and indirect bosses Jos, Sjouke, Wan and Erik, for their support.

I owe a debt of gratitude to Erik. He co-authored the research proposal on which I started my PhD position in Eindhoven, and guided me in the first 18 months. Erik's supervision was honest, engaging, involved, yet stern when necessary – and necessary it was at times. Although he officially ceased supervising me, to this day he continues to be at the ready with advice, support and constructive criticism.

In line with the BRICKS project, on which I was hired, I regularly met Wan and Taolue at the CWI. Those were always stimulating meetings, which I greatly enjoyed. Despite Wan's busy schedule, he never hurried our meetings, and he was always ready to support me.

In Eindhoven, I shared an office with Cas. He showed me how much joy research is. Sharing an office with him was a great pleasure. After about a year and a half, Sjouke took over my supervision. Soon after, he was offered a position in the University of Luxembourg and accepted. He also invited me to come along, which I ended up doing.

In Luxembourg, Sjouke started a new security group with Saša and me. I shared my new office with Saša. While our interaction was incomparable to the interaction between Cas and me, the warmth and pleasure of it were of a similar level. Within a week, we were joined by Baptiste, who is always willing to answer my questions on French, no matter how difficult. The group expanded further and within a short period has become a cohesive group, not in the least due to Sjouke's way of leading the group.

My colleagues in Eindhoven and Luxembourg provided a friendly, interesting and interested environment. Lunches, tea breaks and the yearly outing with Jos, Michiel, Jing, Tijn, Francien, Erik de V. and Erik L., Ruurd, Ronald, Rob, Pieter, Bas, Simona, Sonja, Jasen, Ana, Sonja and Paul of the FM group were always great. Lunches, tea breaks, dinners (group barbecues!) and the occasional classical concert with Sjouke, Saša, Pieter, Ton, Jun, Chenyi, Barbara and Wojtek were equally animated.

During my PhD thesis, I had the pleasure to write together with Wolter, Mohammad, Srijith, and Melanie. Collaborations with them were enjoyable, and fun, and frequently led to after-hours discussions. I look forward to collaborate with all of you again in the future. In addition, discussions on the research often popped up, with various colleagues sharing their insights with me. This thesis incorporates suggestions

of Zhe Xia (the attack on ALBD04), Ton van Deursen (the readability predicate), Saša Radomirović (list propagation), and some of the results of various discussions with Peter Ryan on Prêt à Voter .

At various workshops, meetings and visits, I have met, collaborated and enjoyed discussions and informal chats with many people, of which I will here name Josh Benaloh, Rop Gonggrijp, Leontine Loeber, Piet Maclaine Pont, Jens Calamé, Anton Wijs, and the IPA-members (especially the PhD council, Tijn and Sonja). Thanks for your insightful comments, and the way we often livened up meetings.

The fact that I haven't lost touch with my friends, even after having emigrated twice, best describes their support. I am grateful for the friendship of the HoopjePlus'ers, the Horsepower-group, Micha & Suzanne, Marek & Miranda, Mathieu, Marjan, Daniel and Tessa. Ward and Ania both deserve specific mention: their friendship and the example they set, gave me the confidence to emigrate.

A special mention goes to Sjouke. He has supported me throughout my PhD career, has given me the room to explore, provided advice and feedback, often shared a cup of tea with me, and believed in me. I am lucky to have worked with him, and I look forward to doing so again.

Last, but certainly not least, I would like to thank my parents, my sister, my brother and Maren, for their unconditional love and support.

Contents

Acknowledgments	i
1 Introduction	1
1.1 Understanding security	1
1.1.1 Privacy in Luxembourgian elections	2
1.1.2 Tit-for-tat on Ebay	2
1.1.3 Privacy vs. fairness	3
1.2 Approach	4
1.3 Thesis organisation	4
I Privacy in Voting	7
2 Introduction to voting	9
2.1 Domain analysis	9
2.1.1 Establishing stakeholders	11
2.1.2 Core voting system security requirements	11
2.2 Terminology in literature	13
2.3 Conclusions	16
3 A survey of voting schemes and systems	17
3.1 Cryptographic support for privacy in voting	17
3.1.1 Properties of cryptographic systems	17
3.1.2 Proofs	21
3.1.3 Secret sharing	22
3.1.4 Mixnets	24
3.2 Voting schemes and systems	26
3.2.1 Receipt-free voting schemes	27

3.2.2	Practical and end-to-end verifiable systems	40
3.3	Summary	49
4	Formalising voting systems	51
4.1	Related work	51
4.1.1	Privacy expressed in applied π	52
4.1.2	Privacy expressed in logics	52
4.1.3	Approach	53
4.2	Framework for modelling voting systems	53
4.2.1	Syntax of the framework	54
4.2.2	Semantics of the framework	60
4.3	Privacy in voting systems	65
4.4	Conspiring voters	67
4.5	Modelling conspiratory behaviour	68
4.6	Integrating privacy primitives	72
4.6.1	Properties of cryptographic systems	72
4.6.2	Shared secret decryption	74
4.6.3	Integrating proofs	75
4.6.4	Summary	78
4.7	Discussions	78
4.7.1	Privacy from voting authorities	78
4.7.2	Receipt-freeness	78
4.7.3	Coercion-resistance	78
4.8	Conclusions	80
5	Application	81
5.1	Determining the privacy of FOO92	81
5.1.1	FOO92 description	81
5.1.2	Modelling FOO92 in the framework	82
5.1.3	Privacy of FOO92	85
5.1.4	Discussion	88
5.2	Voter-controlled privacy of Prêt à Voter	89
5.2.1	Description of PaV	90
5.2.2	Modelling PaV	92
5.2.3	Voter control of privacy in PaV	95

5.2.4	Discussion	96
5.3	Privacy and verifiability	97
5.3.1	Privacy and verifiability in 3BS	98
5.4	Conclusions	101

II Fairness in Digital Exchange 103

6 Introduction to DRM 105

6.1	Introduction	105
6.2	Domain analysis	107
6.2.1	Establishing stakeholders and their incentives	107
6.2.2	Security properties per core role	109
6.2.3	Conceptual process model	112
6.2.4	Results of the problem analysis	113
6.3	Core security requirements	114
6.3.1	Security requirements for the content creator	114
6.3.2	Security requirements for the distributor	114
6.3.3	Security requirements for the user	116
6.3.4	Consequences	118
6.4	Summary	118

7 Fair exchange in DRM 121

7.1	Introduction	121
7.1.1	Approach to study feasibility	122
7.1.2	Related work in fair exchange	123
7.1.3	Structure of this chapter	123
7.2	Assumptions and notations	123
7.3	The NPGCT DRM scheme	124
7.3.1	NPGCT protocols	124
7.3.2	Results of formally analysing NPGCT	125
7.4	The Nuovo DRM scheme	126
7.4.1	Nuovo's goals	126
7.4.2	Assumptions of Nuovo DRM	127
7.4.3	Nuovo DRM protocols	128
7.5	Formalisation of Nuovo DRM	130

7.5.1	Modelling language	131
7.5.2	System model	132
7.5.3	Formalisation of Nuovo's goals	136
7.6	Analysis results	138
7.6.1	Honest scenario	139
7.6.2	Dishonest scenario	140
7.7	Nuovo DRM mitigation procedures	141
7.7.1	Resolving C2C disputes at the TTP	142
7.7.2	Detection of compromised devices	143
7.7.3	Isolating compromised devices	145
7.8	Summary	148
III	Concluding remarks	149
8	Conclusions	151
8.1	Privacy in voting	151
8.1.1	Future work	152
8.2	Fairness in digital exchange	153
8.2.1	Future work	154
	Bibliography	155
	Index of subjects	167
	Summary	169
	Samenvatting	171
	Curriculum Vitae	175

List of Figures

1.1	Influencing voters	1
1.2	Luxembourgian ballot in Centre region	2
1.3	Fairness dilemma in on-line trading	3
2.1	Individual preferences	10
3.1	Blind signatures	20
3.2	Secret sharing	23
3.3	Mixnet	25
3.4	Development of voting systems	26
3.5	Vote casting in <i>FOO92</i>	28
3.6	Vote casting in <i>CGS97</i>	28
3.7	Voting in <i>SK95</i>	31
3.8	Change in vote casting from <i>SK95</i> to <i>HS00</i>	32
3.9	Extensions to <i>CGS97</i> for receipt-freeness in <i>LK00</i>	33
3.10	Extensions to <i>CGS97</i> for receipt-freeness in <i>MBC01</i>	35
3.11	Per-voter candidate lists in <i>RIES</i>	42
3.12	A ballot in <i>Prêt à Voter</i>	43
3.13	A <i>Punchscan</i> ballot voting for “south”	44
3.14	<i>Punchscan</i> ballot orderings	44
3.15	Vote verification in <i>Scantegrity</i>	45
3.16	A ballot in <i>Scratch & Vote</i>	46
3.17	A <i>Threeballot</i> in favour of “East”	47
4.1	A labelled transition system	65
4.2	Knowledge sharing	68

5.1	The <i>FOO92</i> scheme	82
5.2	A PaV-ballot voting for East	90
5.3	Randomised partial auditing of one Mixer	91
5.4	Elections with PaV	92
5.5	Model of voter interaction in PaV	93
5.6	A <i>Threeballot</i> in favour of “East”	98
5.7	Faking conspiracy in 3BS	100
6.1	Basic process model	112
6.2	Model incorporating rendering of content	113
6.3	Generic process model of DRM	113
6.4	Objective tree for property c1	115
6.5	Objective tree of properties d1 and d3	116
6.6	Objective tree for property u1	117
6.7	Objective tree for properties u2 and u3	117
7.1	Nuovo P2C exchange	128
7.2	Nuovo C2C exchange	129
7.3	Nuovo C2C recovery	130
7.4	Abbreviated syntax of μ CRL process terms	132
7.5	Nuovo specification (P2C)	132
7.6	Nuovo specification (C2C)	133
7.7	Nuovo specification (C2C recover)	134
7.8	Nuovo specification (intruder)	136
7.9	μ -calculus syntax	136
7.10	Use case: one node in a network	147

List of Tables

2.1	Voting stakeholders	11
2.2	Core requirements for voting system	13
3.1	Claims of classical and receipt-free schemes	38
3.2	Usage of cryptographic techniques in voting schemes	39
4.1	Example: matching and substitution of terms	56
4.2	Example: voting systems	59
4.3	Example: reinterpretation of terms	66
4.4	Example: event transformation	70
5.1	Validity of various possible Threeballots	100
6.1	Core stakeholders for DRM systems	107
6.2	Security properties as desired per role	111

1

Introduction

This chapter is intended to sketch the context of the research detailed in this thesis for a reader unfamiliar with security. It introduces the setting, and provides an outline of the thesis.

1.1 Understanding security

Security matters.

In the real world, there are various mechanisms that provide security. For example, valuables are locked in safes, passports identify people crossing borders and signatures on contracts express the agreement of the signing parties. These basic security mechanisms have their equivalents in the digital world. The safekeeping of secrets is called confidentiality, and verifying the correctness of documents such as passports and contracts is called integrity in the digital world.

These basic security concepts have been studied in great depth. Our understanding of these security concepts has matured. In contrast, the understanding of more complex security concepts has yet to progress. In this thesis, we examine two more complex security notions: privacy in voting and ‘tit-for-tat’ in trading.



(a) vote-buying



(b) coercion

Figure 1.1: Influencing voters

In voting, the link between a voter and her vote must remain private, irrespective of any pressure on the voter. In trading, both seller and buyer require ‘tit-for-tat’: neither wants to give without getting something back. These are both complex issues, as we illustrate below.

1.1.1 Privacy in Luxembourgian elections

Privacy in voting relies on the secrecy of the ballot. As no one can link any given ballot to any voter, the choice that the voter made remains secret. However, as Figure 1.1 illustrates, voters may be pressured into revealing this link. We illustrate one way in which a voter can do so in actual elections: the Luxembourgian elections for the *Châmber vun Députéirten* (the parliamentary chamber in Luxembourg).

In the elections for the Châmber, Luxembourg is divided into four regions: North, East, South and Centre. Each region directly selects representatives for their portion of the Châmber: for example, the Centre region elects 21 representatives. Candidates are grouped by party, and the number of candidates per party must be equal to the number of seats in the region. Hence, in the Centre region every party has precisely 21 candidates. To vote, each voter in the Centre region picks 21 preferences, and marks them on the ballot, at most two per candidate (see Figure 1.2).

1. ADR	...	7. KPL
1-1. Jacques Henckes ○ ○	...	7-1. Paul Back ○ ○
⋮	⋮	⋮
1-21. Fernand Zeutzius ○ ○	...	7-21. Marco Tani ○ ○

Figure 1.2: An example of a Luxembourgian ballot in the Centre region

Suppose a vote buyer makes a voter an offer the voter just cannot refuse. Then, the voter wishes to prove how she voted to the buyer. Before the elections, she agrees with the buyer on a specific, unique way to fill in her ballot. If a ballot marked in this way turns up in the set of cast ballots, the buyer knows the voter complied. In the example of Figure 1.2, the voter can fill in $7 \times 21 \times 2 = 294$ boxes. If she votes twice for one specific candidate, she can fill in 19 of the remaining 292 boxes. She has an extremely large number of different, unique ways to vote twice for one candidate (to be precise, $\binom{292}{19} = 314,269,098,408,967,151,724,980,483,800$ possible ways). Consequently, even in a very large group of voters, every voter can fill in the ballot in their own unique way. The ballot then acts as a *receipt*, a proof of how a voter voted.

Receipts are not only an opportunity for voters to sell their votes (Figure 1.1a). Since any voter can produce a receipt, a Luxembourgian Mafia (should such an organisation exist) could *coerce* voters (Figure 1.1b), that is, force voters to produce a receipt. In this way, the Mafia could check each voter to see if she voted for the Mafia's favoured candidate.

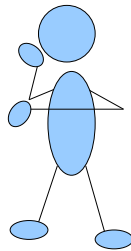
The complexity of such privacy problems cannot be captured fully in terms of confidentiality and integrity – it extends beyond them.

1.1.2 Tit-for-tat on Ebay

Suppose person *A* is looking for an item on Ebay, and happens to find person *B* offering precisely what *A* is looking for. Naturally, *A* bids. If *A* wins, *A* will want to be assured that he will receive the goods *before* sending the payment. Conversely, *B*

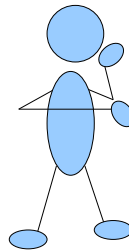
will want assurance that he will receive the payment *before* sending the goods. Both want to finish the trade, but both require assurance that the other will deliver in order to continue. This leads to the dilemma illustrated in Figure 1.3: who goes first?

“Should I pay before
receiving the goods?”



A

“Should I send the goods
before receiving payment?”



B

Figure 1.3: Fairness dilemma in on-line trading

Whoever acts first, risks that the trading partner will receive, but not deliver. Ensuring this is not possible is called *fair exchange*. There are various ways to achieve this with physical items (e.g. books), such as meeting in person or using an escrow service.

However, such options cannot be straightforwardly extended to digital items (such as MP3 files). Transfer of a digital item is done by copying the item. While the transfer procedure could erase the source file, there is no way to guarantee that all copies of the source were erased.

Digital items can be copied effortlessly for free. If an unlimited amount of copies are available, the copies themselves lose all value, and there is no point in trading them. Thus, trade in digital goods can only succeed if there is a limit to the copyability of the goods.

This contrasts with the absoluteness of confidentiality and integrity: either they are satisfied, or not. A qualified requirement such as a limit on copyability is a different matter.

1.1.3 Privacy vs. fairness

This thesis sets out to investigate the two issues above.

A better understanding of privacy in elections is needed to ensure that a voting system does not offer any leakage of privacy (other than the loss of privacy due to publication of the result). To this end, we formalise the notion that anything that reduces privacy is a receipt in the first part of this thesis.

In the second part of this thesis, we investigate how to enable fairness in digital exchange, that is, how to enable limited reselling of digital objects, while preventing unlimited copying.

1.2 Approach

The goal of this section is not to describe the approach to studying the two problems in full detail, but to provide an outline of the approach. In later chapters, the approach is described in a more detailed fashion.

We believe that security objectives cannot be studied in isolation. Security objectives are embedded in a context with other security objectives, a system, users of the system and a malevolent entity, called the intruder, who seeks to upset the system's objectives.

To this end, we begin by analysing the problem domain: voting and digital rights management, which enables trade of digital goods, respectively. From both analyses, there emerges an understanding of parties involved, a set of core security requirements and an understanding of how the core security requirements relate to each other and the parties involved.

The analysis of voting will indicate a consensus on terminology for security objectives, but this terminology is not always sufficiently precise. In the case of privacy, the consensus has not matured into a full understanding of privacy in voting. In digital rights management we do not even find a clear-cut consensus on the terminology for desired security objectives.

In both cases, however, we continue with an investigation of literature to bring the current understanding of privacy in voting and fairness in digital rights management (respectively) to light. In both cases, we will find that there exist systems whose claims are not valid. We will argue that a structured, rigorous, unambiguous approach to security is necessary. The area of formal methods offers this, and we will use formal methods in both cases.

For privacy in voting, there are various systems proposed with claims of privacy, some of which have been broken. We will introduce a quantified formalisation of privacy with which the privacy offered by these systems can be measured. This formalisation will enable us to detect any loss of privacy.

For fairness in digital exchange, there is no vast body of literature available. On the contrary, we find only few works in this area. Thus, we set out to prove the feasibility of fairness in digital exchange by introducing a new digital rights management system, Nuovo DRM. We will formalise the goals of Nuovo DRM and we will verify that Nuovo DRM achieves its goals.

The study of privacy in voting concludes with an application of our formalisation in a case study. The study of fairness in exchange is similarly concluded by a verification using model-checking. As such, both approaches end with substantiating the found results via a formal analysis.

1.3 Thesis organisation

The rest of this thesis is organised in three parts.

Part I: Privacy in Voting

This part of the thesis begins in Chapter 2, by investigating the domain of voting. From this analysis, we derive a set of core requirements for voting systems. By comparison to requirements terminology in existing literature, we see that there is confusion on privacy in voting. The rest of this part of the thesis endeavors to address this confusion.

To this end, we continue in Chapter 3 by clarifying mechanisms that are used to introduce privacy in voting systems. This is followed up by a survey of voting systems, that focuses on systems designed to ensure privacy. We note that not every system lives up to its promised privacy, often newer systems find privacy flaws in older systems and aim to address them.

In Chapter 4, we propose a new formalisation of voting systems. We then express privacy as a measure on this formalisation. This new definition quantifies privacy, and, as such, can determine partial loss of privacy as well as total loss of privacy. We introduce several models of conspiring voter behaviour, and we define how to compare the privacy of a conspiring voter to the privacy of a non-conspiring voter. The chapter ends by discussing how this newly introduced formalisation of privacy relates to existing notions of privacy and verifiability (a voter's ability to verify the result) in voting. This chapter is based on work together with Sjouke Mauw and Jun Pang [JMP09b, JMP09a].

The voting part of this thesis is concluded with Chapter 5. In this chapter, we provide extensions to the formal framework that capture the mechanisms explained in Chapter 3 that introduce privacy in voting systems. Furthermore, we illustrate how the framework is used by establishing the privacy and resistance to conspiracy of a specific voting system, the FOO92 voting scheme [FOO92].

Part II: Fairness in Digital Exchange

This part of the thesis begins in Chapter 6 by investigating the domain of Digital Rights Management systems. From this analysis, we derive a set of core security requirements for DRM systems. This chapter is based on earlier work with Sjouke Mauw [Jon04, JM07, JM09].

In Chapter 7, we study the feasibility of fair exchange in DRM systems. To this end, we survey fair exchange DRM systems, to find only a very limited number of proposals in literature for peer-to-peer exchange. Upon examination, the first proposal in this field is flawed. We design a new system, Nuovo DRM to address these flaws and provide fair exchange. To substantiate the security of Nuovo DRM, we formalise the goals of the original proposal and verify that Nuovo DRM satisfies these goals using model checking. While the verification substantiates the formal security of Nuovo DRM, this security is based on several assumptions. In the remainder of the Chapter, several strategies are detailed that can mitigate the effects of violations of these assumptions. This chapter is based on work together with Mohammad Torabi Dashti and Srijith Krishnan Nair [TKJ07, TKJ08].

Part III: Concluding remarks

This part consists only of Chapter 8. In this chapter, we review the contributions of this thesis and propose future work.

A note on terminology. In this thesis, both the singular voter (in Part I) and the singular user (in Part II) are referred to using female pronouns (i.e. she uses, her ballot, etc.). Furthermore, to distinguish between a system description found in literature and an implemented system, the former is designated “scheme” while the latter is designated “system”.

Part I

Privacy in Voting

2

Introduction to voting

This chapter investigates the domain of voting, in particular the security problems relevant to it. This analysis sets the stage for the next chapters. The analysis first defines the goal of a voting system and its stakeholders. The definition of the goal of a voting system gives rise to a high-level requirement, which is further refined iteratively. The analysis results in a list of core requirements on voting systems, from which the research question emerges.

Having established the core requirements for voting, we briefly review some security techniques particularly well-suited to the problem domain. The chapter ends with an overview of the setting and notation used in the remainder of this part of the thesis.

2.1 Domain analysis

Throughout history, humans have banded together so as to better face external threats as well as to enjoy mutual benefits. Inevitably, groups need to determine a group preference – from simple issues (follow the bison or stay), to complex issues (how to best address environmental concerns). Voting is, in its most basic form, a way to determine a group preference based on the individual members' preferences.

As an example, consider the preferences of three people Alice, Bob, and Carol. They have to decide where the group will go next, either North or South:

- Alice prefers North,
- Bob prefers South, and
- Carol prefers North.

In this example, we can see that the group as a whole prefers North. However, combining individual preferences is not always straightforward. Consider the following extension to the example (see Figure 2.1): the group is extended with Dave, and the group may go East or West as well as North and South. Suppose that Alice wants to go North, and Carol prefers East, Dave does not have a preference, and Bob prefers South first and West second. Establishing the group's preferred direction given the preferences as depicted in Figure 2.1 is impossible: Bob submitted his preferences in a different format than the others, Dave absented and on top of that, the individual preferences do not determine any one direction as the most preferred direction.

Alice	Bob	Carol	Dave
North	South 1	East	—
	West 2		

Figure 2.1: Example of hard-to-combine preferences.

From here on, we take the view that in seeking to establish a group preference, the group has settled on a standardised format in which each individual preference must be expressed (thus disallowing Bob’s deviation). Furthermore, the group has settled on a way to combine the pronounced individual preferences into a group result. The exact way of combining individual preferences into a group preference can vary from group to group, and even from election to election within a group. However, we assume that the group members are always in agreement before establishing a group preference on how to combine the individual preferences to determine the group preference. As we abstract away from this detail, we consider the group preference to be the set of individual preferences.

Given these assumptions, a voting system is a system that determines the group preference, i.e. the set of all individual preferences. More specifically:

Concept 1 (Voting system). *A voting system is a system that as input takes individual members’ preferences, and produces the group preference.*

We call a voting system *acceptable* if the results it produces are accepted by the group members.

We also account for a party that tries to subvert the elections. A clearer picture of what this attacker can do will emerge from the requirements analysis below .

Requirements on voting systems as found in literature are often derived from many different sources, including legal texts that mandate requirements (see e.g. [LKK⁺03, LGK02]). However, in general, it is unclear how requirements in these various sources have been established. Therefore, in order to arrive at the core requirements for voting systems, we analyse the requirements of the stakeholders.

Terminology. Throughout the rest of Part I of the thesis, the following terms will be used to describe the voting process:

- election the entire process of determining a group preference.
- voter an individual group member, referred to as “she”.
- authorities those parties administering (parts of) an election. Often, at least two tasks are distinguished: voter registration (handled by the Registrar) and counting of votes (handled by the Counter or Tallier).
- voting credentials objects proving the subject is a voter (i.e. allowed to vote).
- option, candidate the items on which a preference is sought.
- vote an expression of an individual’s preference.
- ballot an object in a voting system representing a cast vote.
- to cast to submit a voter’s preference to the voting system.
- to tally to count votes.

2.1.1 Establishing stakeholders

We consider the stakeholders involved in voting systems. On the one hand, there are *voters* – the individuals whose preferences are sought as input. On the other hand, there is a group of individuals administering the process, the voting authorities. We separate these two stakeholders, even though process administration can be fully handled by the voters (for example in the case of leader elections [Lan77]). In such situations, we consider the voters to be the authorities. The stakeholders and their roles are summarised in Table 2.1:

<i>Group</i>	<i>Role</i>
voters	cast votes
authorities	administer the election

Table 2.1: Voting stakeholders

As is evident from Table 2.1, authorities do not cast votes, they serve only to administer the process. As such, any core requirements they desire from the process either follow from the voters' requirements, or are not imposed. In the following analysis this is assumed, and thus the analysis only focuses on voters.

In addition to the identified stakeholders, every voting system is embedded in a context – there are outside parties that have an interest in the result or the process, but are not directly part of the process. Examples of such parties include labour unions, politicians, lobby groups, non-governmental organisations, current government, legislature, and so on, as well as attackers. Most requirements such parties would impose on the voting system follow from requirements voters impose. Note that there are exceptions. Consider, for example, the often-seen lack of anonymity in votes by boards of directors. This lack does not follow from a requirement of the board members, but from the shareholders (a party in the context), who require accountability of the board. Nevertheless, we limit the analysis below to the stakeholder requirements.

The analysis below uses the following methodology to arrive at the core requirements: it starts with the initial high-level requirements of voters, which are then progressively refined until the analysis arrives at core requirements for voting systems.

2.1.2 Core voting system security requirements

The goal of the analysis is to uncover core security requirements for voting systems. This means that other requirements, such as accessibility for disabled voters, are not covered.

Given the notion of an acceptable voting system described above, a voter requires the following of a voting system: the result is a correct transformation of the system's input, and the correctness can be verified.

Thus, the two initial security requirements are stated as follows:

- (IR1) The result represents the group preference correctly.
- (IR2) The result is verifiably correct.

We detail both these requirements further.

Initial requirement 1

Because of the symmetry between group members, the group preference is the unification of the individual voters' preferences. A voting system takes these individual preferences, processes them and produces the group preference – the set of all individual preferences. Thus, for the group preference to be represented correctly, all voters' votes must be correctly treated at each stage, from input to final result, and accounted for only once. Thus, requirement IR1 is detailed further as:

- (R1) All voters are free to cast their votes.
- (R2) Only votes cast by voters are taken into account.
- (R3) Every voter's cast vote is taken into account only once.
- (R4) The result of the system depends correctly on all cast votes.

Requirement R1 uses the term *free*. This freedom means that a voter is not restricted from providing her genuine preference as input. Stated differently, voters need not overcome any external influence in casting their votes. They are unrestricted, “free” to vote.

Note that this does not require voters to cast their votes. Concept 1 left this question open as this is not a defining property of voting systems. Hence, whether abstainment is allowed or not is not a core voting system requirement, and thus we leave this open.

Requirement R4 states that the result depends “correctly” on the individual votes. This means that the result accounts for all individual votes as cast, and that the result depends on nothing more than the cast votes. Thus, requirement R4 is detailed further as follows:

- (R4a) The result accounts for all cast votes.
- (R4b) The result accounts for each vote as cast.
- (R4c) The result depends on nothing but the cast votes.

Initial requirement 2

In refining requirement R4, the term “correctly” was expanded. A similar expansion applies to requirement IR2 (“verifiably correct”), but: voters do not know the entire set of cast votes. If voters could know which votes had been cast, then they know the set. Then, voters do not need a voting system to determine this set. Thus, voters can only verify that their own vote is accounted for correctly, and that the result is based on a list of cast votes (and nothing else). More specifically:

- (R5) Every voter can verify that the result accounts for her cast vote.
- (R6) Every voter can verify that the result accounts for all votes as cast, and on nothing further.

The full list of requirements uncovered in this analysis is reproduced in Table 2.2.

- R1:	all voters are free to cast their votes.
- R2:	only votes cast by voters are taken into account.
- R3:	every voter's cast vote is taken into account only once.
- R4:	the result of the system depends correctly on all cast votes, i.e.:
- R4a:	the result accounts for all cast votes.
- R4b:	the result accounts for each vote as cast.
- R4c:	the result depends on nothing but the cast votes.
- R5:	every voter can verify that the result accounts for her cast vote.
- R6:	every voter can verify that the result accounts for all votes as cast, and on nothing further.

Table 2.2: Core requirements for voting system

The list of requirements in Table 2.2 is derived from Concept 1 of a voting system. Providing a more detailed concept of voting (e.g. accounting for abstainment) will result in a more detailed set of requirements. The trade-off for this increased detail is that such a concept is more narrow, that is, the resulting requirements are less generic in nature. Consequently, while the above analysis does not provide a complete, detailed list of voting requirements, it uncovers the *core* requirements for voting systems. The lack of completeness is due to the fact that the starting point for the analysis is not a fully implemented voting system, but a conceptual notion of voting systems. As such, the uncovered requirements hold for a generic class of voting systems (as opposed to the more complete list an analysis of a fully detailed voting system could have resulted in). Simply put, by refining the initial setting, the analysis and thus the analysis results can be further refined. We conjecture that no analysis found in literature can be considered complete, that is, any analysis found in literature may similarly be further refined by further detailing the setting.

2.2 Terminology in literature

Throughout literature, a diverse terminology has emerged to describe voting requirements. This diversification results from the ad hoc nature in which some new terms are established. Newly found attacks are translated into a new requirement, which receives a name. Later, such requirements are further generalised, and a new term emerges that provides wider coverage than the original term. Moreover, on more than one occasion, one term is interpreted differently by different researchers.

Below, we provide an overview of the most common terms found throughout literature, and indicate some of the inconsistencies in terminology. The goal of the list below is not to provide a complete overview of all terminology in literature, but to make the reader familiar with the common terms and concepts used and show how these terms relate to the core requirements as listed in Table 2.2.

- **eligibility:** only eligible voters (those part of the group) may vote. Eligibility is found e.g. in [FOO92, LBD⁺03, ALBD04]. Eligibility is related to requirement R2.
- **democracy:** only eligible voters may vote, and they may only vote once. Obviously, this term encompasses eligibility (as described above). Systems using eligibility often refer to the second part of the democracy requirement (one vote per voter) under different names. Examples include double-voting [Din01], unreusability [FOO92, LK00], prevention of double-voting [ALBD04]. Democracy is used in e.g. [KW99], and is called “eligibility” in [CGS97]. Democracy is related to requirements R2 and R3.
- **accuracy:** accuracy comprises the following three requirements:
 1. The result depends on *all* cast votes.
 2. The result depends on *nothing more* than the cast votes.
 3. The result depends on the votes *as they were cast*.

Accuracy as described here is used in e.g. [KW99, Din01]. In [FOO92], the term “completeness” is used similarly, except that it does not prevent the result from depending on more than the cast votes. In [LK02], “completeness” only covers the first sub-requirement, and “soundness” addresses the second sub-requirement.

Accuracy is related to requirement R4.

- **universal verifiability:** given a set representing the cast votes, the announced result can be verified by anyone. Universal verifiability is named as such in [SK95, CFSY96, LBD⁺03, CCM08], amongst others. In other works, such as [FOO92, BT94, Oka97, ALBD04], a requirement for “correctness” or “verifiability” or “universally acceptable” is mentioned, but what this term means is not made explicit. The term “universal verifiability” has been deprecated in lieu of more detailed terms (see below). Universal verifiability is related to requirement R6.
- **individual verifiability:** a voter can verify that her vote is counted as she cast it. Individual verifiability is found in e.g. [SK95, CCM08]. In [MBC01], the concept is alluded to as “atomic verifiability”. As with universal verifiability, use of the term individual verifiability has been superseded by the more detailed terms mentioned below. Individual verifiability is related to requirement R5.
- **privacy or vote-privacy:** the system does not reveal which candidate the voter chose. There exist various alternate definitions (for example, variants of “all votes must be secret” [FOO92, KW99, LK00], “vote-voter relationship must be kept private” [ALBD04], indistinguishability of ballots [CGS97]). The fact that there is no clear consensus in literature on a definition of vote-privacy is a clear indication that this notion has not been fully investigated yet. Vote-privacy is related to requirement R1.

- **receipt-freeness:** the voter cannot prove how she voted.
This term was introduced by [BT94], in the context of vote-buying, and has received widespread usage. In [JV06], a receipt is characterised as a piece of knowledge that is only possessed by the voter, identifies the voter's vote uniquely and proves that the vote was cast.
Receipt-freeness is related to requirement R1 as well.
- **coercion-resistance:** The voter cannot be coerced.
This notion was first introduced in [JCJ05], and has since been used erratically. The original definition encompasses receipt-freeness and resistance to the following attacks:
 - randomisation: the voter is forced to vote for a random candidate,
 - simulation: the voter is forced to give her credentials to the attacker, who then votes in her stead,
 - forced abstention: the attacker forces the voter not to vote.

Other works assign a different meaning to this term (or to a similar-sounding term), for example resistance to forcing a voter to vote in a particular way (e.g. [LK02], or even equating it with the above definition of receipt-freeness ([MBC01] and to some extent [CCM08]). Again, the fact that there is no clear consensus clearly indicates an incomplete understanding of this concept.
Coercion-resistance, too, is related to requirement R1.

Note that vote-privacy, receipt-freeness and coercion-resistance all relate to requirement R1. They all embody a different aspect of privacy. There is no consensus in literature on these concepts.

In addition to the terms mentioned above, several new terms emerged recently that deal with aspects of verification by voters. These terms originate from a new direction in research, end-to-end verifiable voting systems. End-to-end verifiable voting systems focus on ensuring that voters can verify each step in the voting process, from putting their vote into the system to verifying the result (see Chapter 3). This is to mimic the *chain of custody*, that is often present in paper-ballot elections. Chain of custody for paper ballots implies that at every stage of the voting process observers (e.g. voters) may be present if they choose, thus ensuring that any tampering with the process can always be observed and, thus, detected.

All the requirements listed below detail aspects of verifiability. Hence, each of these is related to requirement R5 and/or requirement R6. Some of the terms below presuppose the existence of a public record of cast votes. A system may publish such a record; it must do so in order to satisfy a requirement which presupposes the public existence of such a record.

- **cast-as-intended, recorded-as-cast:** in literature, a distinction emerges between correctly transcribing a voter's input and correctly storing all inputs. However, these concepts are fairly new, and consensus on terminology seems to be emerging only now. The terms *cast-as-intended* and *recorded-as-cast*, are being used more and more to describe these respective concepts. This thesis takes

cast-as-intended to mean that the voter can verify that her input is what she intended (e.g. by displaying a dialogue “you chose ..., are you sure [y/n]?” before the voting system considers a vote as having been cast). We take *recorded-as-cast* to mean that the system’s record of ballots includes the unmodified voter’s ballot.

The concept *cast-as-intended* occurs for example in [RS07, CCC⁺08, Rya05, BMR07], while the *recorded-as-cast* concept is found, for example, in [Cha04, CEC⁺08, FCS06, CCC⁺08, AR06].

- **tallied-as-recorded:** anyone can verify that the published record of ballots conforms with the result as announced by the system.
This concept can be found in e.g. [Cha04, CEC⁺08, CCC⁺08, AR06, BMR07, FCS06]. Note that the latter two use the equivalent term *counted-as-recorded*.
- **counted-as-cast:** Any voter can verify that her vote counts in favour of the candidate for which she cast it.
Counted-as-cast combines the ideas of *recorded-as-cast* and *tallied-as-recorded*. This term occurs e.g. in [RS07], while the previous mentioned works allude indirectly to the same concept.

2.3 Conclusions

Voting requirement terminology suffers from inclarities. For some requirements (such as R4) the situation seems not so bad. In other cases, most notably R1, a clear consensus is absent in literature. Note that even requirements on which consensus did exist (e.g. R5, or R6), may suddenly become subject of refinement when a new approach (in this case, end-to-end verifiability) emerges. In the case of R5 and R6, this gave rise to several new terms, leading again to confusion.

The confusion surrounding requirement R1 and the interaction with requirements R5 and R6 (verifiability) indicates that the underlying concepts of these requirements are not well-understood in literature. Privacy is a fundamental concept for requirement R1: if privacy is ensured, then no one can hold a voter accountable for her vote. Although privacy itself is not sufficient to ensure R1 (for example, the election office can be too far off to vote for certain voters, or a candidate may hold all voters accountable for the election result), it is an essential ingredient to ensure freedom to cast votes. However, the confusion surrounding the concept of privacy in voting indicates that privacy is particularly poorly understood. Therefore, the concept of privacy in voting must be clarified. This leads us to formulate the research question for this part of the thesis as follows:

Research question I. *Investigate and improve understanding of privacy in voting.*

We address this question as follows. First, we survey voting systems from literature to investigate how privacy is addressed by these systems in Chapter 3. The survey enables us to pinpoint the nature of the confusion surrounding privacy in voting. This information is used to guide the development of a privacy framework for expressing voting systems in Chapter 4. The framework is applied to several voting systems from literature in Chapter 5.

3

A survey of voting schemes and systems

The previous chapter established core voting system requirements, and highlighted that there is no clear consensus in literature on several of these core requirements. This chapter provides a survey of the status of requirements R1 (privacy) and R5 and R6 (verifiability) in computer science research. First, some standard techniques from cryptography that support privacy and verifiability in electronic voting systems are explained. Then, this chapter surveys voting schemes and systems with a focus on privacy and verifiability.

Note that the discussion is not intentionally focused on electronic voting systems. However, as the discussed systems all originated from computer science research, many of them are electronic voting systems and thus the survey, by necessity, mostly details electronic voting systems.

3.1 Cryptographic support for privacy in voting

As remarked in the previous chapter, the interaction between privacy (requirement R1) and verifiability (requirements R5,R6) is not well understood. Several cryptographic concepts have been introduced that enable electronic voting systems to provide privacy whilst preserving verifiability. The concepts discussed are those prevalent in current voting systems: privacy-enhancing encryption techniques, mixnets, secret sharing, and proofs.

3.1.1 Properties of cryptographic systems

Cryptographic systems may exhibit properties that support privacy. Such properties can obviously be leveraged for voting systems. Below, we discuss three such properties: homomorphic encryption, blind signatures and re-encryption. This is followed up by a note on the security implications of cryptographic systems which exhibit such properties.

A note on notation. In the setting of this thesis, cryptography operates on strings of bits. To explain the working of the various properties, the properties are provided with a signature expressing their working. This signature uses B to denote any string

of bits. Furthermore, we distinguish between various subclasses of bitstrings. For example, encryptions are denoted B_{enc} , signatures are denoted B_{sig} , and keys as $Keys$. As an example of notation, we provide the descriptions of encryption and signing.

- Encryption takes a bitstring and a key, and outputs an encrypted bitstring. Encryption is denoted by $\{-\}_-: B \times Keys \rightarrow B_{enc}$.
- Signing takes a bitstring and a key, and outputs a signed bitstring. Signing is denoted by $sign_-: Keys \times B \rightarrow B_{sig}$.

Homomorphic encryption

The main difference between homomorphic encryption and regular encryption is that in homomorphic encryption, there is an algebraic operation on multiple ciphertexts that results in an algebraic operation on the corresponding plaintexts. The specific algebraic operations depend on the used cryptographic system.

Notation. We use B_{HE} to denote bitstrings representing homomorphically encrypted messages. Homomorphic encryption concerns the following three functions.

- $\{\!\{ \}$, which encrypts a message in B . It has the following signature: $\{\!\{-\}_-: B \times Keys \rightarrow B_{HE}$.
- \otimes , which denotes an appropriate operation on encrypted messages. It has the signature $_- \otimes -: B_{HE} \times B_{HE} \rightarrow B_{HE}$.
- \oplus , which denotes the appropriate operation on messages. It has the signature $_- \oplus -: B \times B \rightarrow B$.

Properties. Homomorphic encryption is characterised as follows. Given two messages ma, mb , and a key k , we can produce $\{\!\{ma\}\!\}_k$ and $\{\!\{mb\}\!\}_k$. Homomorphic encryption shares the following properties of regular encryption:

- Neither m nor k can be learned from $\{\!\{m\}\!\}_k$ without the decryption key k^{-1} .
- $ma \neq mb \implies \{\!\{ma\}\!\}_k \neq \{\!\{mb\}\!\}_k$.
- $ka \neq kb \implies \{\!\{m\}\!\}_{ka} \neq \{\!\{m\}\!\}_{kb}$.

Regular encryption has an additional property:

- $\{\!\{m\}\!\}_k$ cannot be made without key k and message m .

In homomorphic encryption, this property is relaxed somewhat. This is due to the homomorphic property, which allows two encrypted messages to be combined meaningfully (using \otimes) into one encrypted message without using a key:

$$(HE1) \quad \{\!\{ma\}\!\}_k \otimes \{\!\{mb\}\!\}_k = \{\!\{ma \oplus mb\}\!\}_k.$$

The actual operations represented by \otimes and \oplus depend on the used cryptographic system. Examples of homomorphic cryptographic systems include RSA [RSA78] (where both \otimes and \oplus are multiplication), ElGamal [Elg85] (again, both \otimes and \oplus are multiplication), and Paillier [Pai99] (where \otimes is multiplication and \oplus is addition). A detailed survey of homomorphic systems is given in [FG07].

Application in voting. Homomorphic encryption can be applied to tally votes using asymmetric cryptography. In asymmetric cryptography, the decryption key is different from the encryption key. Furthermore, it is believed to be intractable to determine the decryption key given the encryption key.

In such a setting, every voter can encrypt her vote with the election public key. The authorities add all votes together, which results in the encrypted tally. If \bigotimes_{set} denotes application of \otimes to each element of set , and \bigoplus_{set} similarly denotes application of \oplus , then summing up all the votes $\gamma(v)$ by the voters $v \in \mathcal{V}$ is written as:

$$\bigotimes_{v \in \mathcal{V}} \llbracket \gamma(v) \rrbracket_k = \llbracket \bigoplus_{v \in \mathcal{V}} \gamma(v) \rrbracket_k.$$

Anyone can verify that this equality holds, but only the voting authorities can decrypt the result. Thus, any voter can verify that her vote contributes to the result, but no one except the authorities can see for which candidate a particular voter voted. This preserves vote-privacy while offering verifiability. In combination with secret sharing (see below), homomorphic encryption can even prevent the authorities from learning for whom a particular vote was cast.

Blind signatures

A blind signature [Cha82] is a way for a party to sign a message without knowing the message. Compare this to a sealed envelope, containing a piece of carbon paper and a message to be signed. The carbon paper will copy anything written on the envelope to the message, including a signature. Thus, a notary can sign the envelope, while the secret message is not revealed to him. The envelope (the blind) can be removed, revealing a signed message.

Notation. We denote a blinded message with B_{blind} . In blind signing, the following two functions are used:

- $\llbracket \cdot \rrbracket$, which blinds a message.
 $\llbracket \cdot \rrbracket_- : B \times Keys \rightarrow B_{blind}$.
- *deblind*, which removes the blinding from a signed blinded message.
 $deblind : B \times Keys \rightarrow B_{sig}$.

Properties. If a blinded message is signed, the message can be debinded without removing the signature. For a message m , blinded with key bk , signed with key $sk(a)$, we have the following.

$$(BS) \quad deblind(sign_{sk(a)}(\llbracket m \rrbracket_{bk}), bk) = sign_{sk(a)}(m).$$

Application in voting. Blind signatures can be used to ensure that only eligible voters can vote (see Figure 3.1). A voter v blinds her vote $\gamma(v)$ using blinding key bk . She signs that and sends her signed, blinded vote to the registrar. The registrar separates the blinded message and the signature, and verifies that the voter is eligible and has not yet voted. If so, the registrar signs the blinded vote and sends it back to the voter. The voter unblinds the received message using bk and sends the signed vote to a tallier.

$v \rightarrow \text{registrar:}$	$\text{sign}_{sk(v)}(\llbracket \gamma(v) \rrbracket_{bk})$
$\text{registrar} \rightarrow v:$	$\text{sign}_{sk(R)}(\llbracket \gamma(v) \rrbracket_{bk})$
$v \rightarrow \text{tallyer:}$	$\text{sign}_{sk(R)}(\gamma(v))$

Figure 3.1: An example of the use of blind signatures.

Re-encryption

Re-encrypting a message means changing the ciphertext, without changing the corresponding plaintext. Knowledge of the decryption key is not needed for re-encryption.

Notation. We denote re-encryptably encrypted ciphertext as B_{renc} . A re-encryptable encryption of a message m has two parts inside the encryption: the message m and a random factor r . We denote the set of random factors as $Nonces$. Using this, re-encryption of message m is written as follows.

- $\langle \{ | \} \rangle$, which encrypts a message in a form suitable for re-encryption.
 $\langle \{ - | - \} \rangle_- : B \times Nonces \times Keys \rightarrow B_{renc}$
- $rcrypt$, which re-encrypts a message.
 $rcrypt(-, -) : B_{renc} \times Nonces \rightarrow B_{renc}$
- f , a function that combines two random factors.
 $f(-, -) : Nonces \times Nonces \rightarrow Nonces$

Properties. A re-encrypted ciphertext is distinct from the original ciphertext. However, both decrypt to the same plaintext. Thus, for message m , key k and randoms r, r' we have the following:

$$(RE1) \quad rcrypt(\langle \{ m | r \} \rangle_k, r') = \langle \{ m | f(r, r') \} \rangle_k$$

$$(RE2) \quad \langle \{ m | f(r, r') \} \rangle_k \neq \langle \{ m | r \} \rangle_k$$

Examples of cryptographic systems allowing re-encryption include ElGamal [Elg85], Goldwasser-Micali [GM84], and Paillier [Pai99].

Application in voting. In voting, re-encryption is used to break the link between a voter and her submitted vote. By re-encrypting a cast vote, the vote remains unchanged, while the encrypted message no longer matches the message the voter cast. An extension of this is used in mixnets (which are explained below), to break the link between the input and the output of a mixnet.

Security implications

Note that each of the above properties allows a ciphertext to be altered in a meaningful way without access to the decryption key. Cryptographic systems with this property are called *malleable* [DDN00]. Malleability of the cryptographic system can be leveraged for privacy (because a ciphertext can be changed by someone not knowing the plaintext, i.e. by someone other than the encrypting individual). However, a malleable ciphertext can be changed in a meaningful way by an attacker as well. Due to this, malleable cryptographic systems do not achieve the highest levels of cryptographic security (ciphertext indistinguishability under adaptive chosen ciphertext attack, IND-CCA2). Despite this, such systems can attain a certain amount of security (e.g. indistinguishability under chosen plaintext attack, IND-CPA). As the ciphertexts of a malleable cryptographic system can be changed by anyone in a meaningful way, such systems must be used with care and awareness of their limitations and vulnerabilities.

3.1.2 Proofs

A proof is used to convince another party of the correctness of a certain statement. For example, consider a proof that one message m' is a re-encryption of another messages m . This can be proven in various ways, e.g. by revealing the random factor used to re-encrypt.

Proofs can have various properties. We discuss two properties that ensure that the proofs do not reduce privacy: zero knowledge proofs and designated verifier proofs.

Zero knowledge proofs

A zero knowledge proof (ZKP) is a proof of a statement s which reveals nothing about statement s , except its truth. For example, the statement $\exists r: m' = \text{rcrypt}(m, r)$ reveals that m' is a re-encryption of m , but not how. A proof of this statement that only proves that it is true, but does not reveal r , is a zero knowledge proof.

Observe that in this example, both prover P and verifier V know the statement $\exists r: m' = \text{rcrypt}(m, r)$ and m and m' . However, only P knows the value of r (V knows some value is used, but not which, and the statement does not reveal the value). As shown by the example, a zero knowledge proof deals with some public knowledge and some private knowledge. More specifically: a zero knowledge proof proves a boolean statement s over publicly and privately known arguments, without revealing the private knowledge.

Application in voting. Zero knowledge proofs are often used in voting to prove to a voter how her vote is being processed by the system. This enables a voter to verify that her vote is being handled correctly, without her being able to prove how she voted. An example of this is proving correctness of encryption without revealing the encryption key. This is done in mixnets (see below).

Designated verifier proofs

Regular proofs are transferable. A voter who possesses a proof of a boolean statement s can transfer that proof to anyone. This is not always desirable, for example a voter selling a proof of how she voted. Suppose prover P wants to prove the truth of statement s to verifier V , but doesn't want V to be able to prove s to anyone else. What P can do, is prove the following to V :

“I am V OR s .”

As the verifier knows that the prover is not V , the statement s must be true. As “I am V ” holds true for the verifier, the verifier can prove “I am V OR x .” for any x (by proving that the verifier is V). Hence, V cannot use the above proof to convince anyone else of statement s .

Application in voting. Designated verifier proofs are used much like zero knowledge proofs. They enable the possibility of revealing information to a voter that enables her to verify that her vote is counted. Since these proofs cannot be forwarded, this does not impact the voter's privacy. As for zero knowledge proof, a possible application is their use in mixnets, e.g. shuffling a list of candidates and revealing the order only to the voter [HS00].

3.1.3 Secret sharing

There are instances where access to information is deemed too dangerous to reside with one person. A classical Hollywood example is firing a nuclear weapon: this often requires two keys, held by two different individuals. Cryptographers have devised a mechanism to achieve a similar result cryptographically [Sha79, Bla79]. A secret is shared by a number of authorities, and only by putting the shares together, can the authorities recover the secret.

A technical solution works by interpolation of polynomials (see Figure 3.2). A polynomial of degree n is uniquely determined by any $n + 1$ points of the polynomial. Hence, given $n + 1$ points, a polynomial can be uniquely determined.

To share a secret, the secret is encoded as a number m . Then, a random polynomial is chosen that covers the point $(0, m)$. Random coordinates on the polynomial are then chosen as shares of the secret. Each share is then given by a random point on the polynomial.

For example, suppose that a secret message m is shared over four shares. This requires a polynomial of degree 3 (number of shares minus one). In Figure 3.2, the secret $m = 1$ is shared over four shares s_0, \dots, s_3 using the polynomial $-0.2x^3 + 0.5x^2 + 0.7x + 1$.

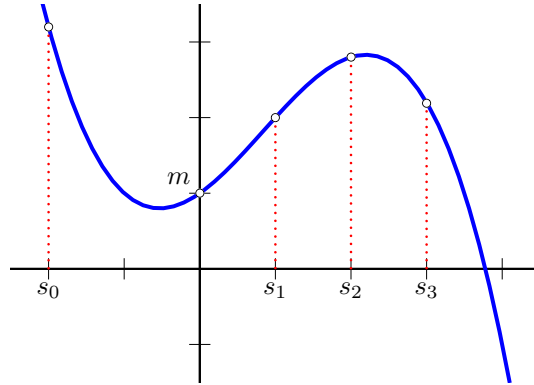


Figure 3.2: Secret m shared over s_0, s_1, s_2 and s_3 .

To recover the secret, the four points given by the shares are interpolated, which reconstructs the original polynomial. The secret message is then given by the value of the interpolated polynomial for $x = 0$.

Note that the benefit of secret sharing comes at a price: to recover the secret, *all* shares are required. One missing share makes it impossible to recover the secret. This means that any one unwilling party, who holds a share, can prevent recovery of the secret.

In order to mitigate this, more shares can be handed out. As a polynomial of degree n is uniquely determined by *any* $n + 1$ points of the polynomial, any desired number of shares can be handed out. The polynomial can be reconstructed using any subset of $n + 1$ of these shares. This is called *threshold cryptography*. Often, a secret sharing scheme is referred to as (t, n) meaning that t (for threshold) shares are needed, and that there are n shares in total. As threshold cryptography prevents the problem of a single share holder blocking attempts to recover the secret, it is often used when secret sharing is applied.

In general, there are two types of secrets which can be protected by secret sharing.

- secrets too valuable to trust to one person (e.g. the launch code for a nuclear weapon).
- secret inputs to computations (e.g. a decryption key).

While secrets of the first type are meant to be reconstructed when necessary, this is not the case for secrets of the second type. Indeed, it would be better if the computations could proceed without reconstructing the secret – in this way, the secret would remain protected. This is possible in some cryptographic schemes, such as ElGamal [Elg85]). In ElGamal, an asymmetric cryptographic scheme, it is not necessary to reconstruct the secret key in order to decrypt an encrypted message. As an example of how computations can be achieved using shares instead of using the secret itself, we sketch how shared-key decryption is done in ElGamal (a public-key, homomorphic encryption scheme).

Example. Encryption in ElGamal works as follows. The public key is given by raising the generator g of a given group to the private key k : g^k . Encrypting message m results in $(g^r, (g^k)^r m)$, where r is chosen by the encrypting agent. To decrypt, one raises the first element g^r to the power k , which results in the term g^{rk} . Then, the second element $(g^k)^r m$ is divided by g^{rk} , resulting in m . Note that the private key k is not needed for decryption. Decryption only requires g^{rk} , which is derived from k .

A shared decryption of $(g^r, (g^k)^r m)$ is done as follows. Assume the private key k is split into n shares s_1, \dots, s_n , such that $g^k = g^{s_1+s_2+\dots+s_n}$. Each shareholder i raises g^r to his share and sends $(g^r)^{s_i}$ to the others. Now, the shareholders can compute $\prod_i (g^r)^{s_i} = (g^r)^k$, which is the term needed to decrypt $(g^r, (g^k)^r m)$. In this fashion, the message can be recovered without constructing k .

Application in voting

Secret sharing can be used in voting to distribute a decryption key. Suppose all voters encrypt their votes with the same public key. The corresponding private key is then split over all authorities, to ensure that only the authorities acting in concert can uncover the votes. As explained above, it is not even necessary to reconstruct the private key to decrypt the votes. In combination with homomorphic cryptography, the encrypted result could be established, which is then decrypted. In this process, the private key is not reconstructed, thus ensuring that no authority can decrypt any other message.

Another use of secret sharing allows a voter to share her vote over several authorities. This requires that an authority can combine all the shares he receives from the voters. For example, suppose there are only two candidates and two voting authorities. The candidates can be encoded as $\{-1, 1\}$. Each voter splits her vote into two random shares. The sum of these shares is either 1 or -1 and encodes the vote. A voter wishing to vote 1 could have a share 539 and another share -538 . Another voter could have shares 15 and -16 for voting -1 . Now, each voter sends one share to each authority. Thus, the first authority receives (for example) 15 and -538 , and the second -16 and 539. Note that neither authority can determine how any voter voted. Next, each authority sums the received shares (resulting in -523 and 523). The authorities add their sums together, resulting in 0. In this case, both candidates got an equal number of votes, and the authorities did not learn how any voter voted.

3.1.4 Mixnets

Chaum [Cha81] proposed a cryptographic way to shuffle lists. A mix shuffles its input in such a way that no element of the output can be linked back to an element in the input. To ensure that the link between input list and output list is not trivially retrieved, the mix needs to change the appearance of every message *without* changing the meaning of any message. Using cryptography, this can be achieved by decrypting or re-encrypting the messages. To decrypt a message, the mix must possess the decryption key. To this end, the input messages are encrypted with the public key of the mix. Re-encryption does not require the mix to possess a decryption key, and thus puts less restrictions on the input. To shuffle a list, a mix decrypts or re-encrypts each element, and randomises the order in which the elements appear in the list.

In order to ensure that no single entity can recover the ordering, multiple mixes can be chained to form a mixnet. This is possible with both decryption mixes and re-encryption mixes. In the case of decryption mixes, each mix removes one layer of encryption. The initial input must then consist of several layers of encryption, each layer specifically targeted at one mix. The last mix in the mixnet then outputs the finalised list. This approach is used in onion routing [SGR97], which provides anonymous routing. With re-encryption mixes, the initial overhead of encrypting each message once for each mix is avoided.

Figure 3.3 sketches the working of a mixnet. The first mix takes an input list of messages (here, w, x, y, z) and outputs a shuffled list (x, w, y, z) in such a way that the individual messages cannot be linked to its input (denoted in Figure 3.3 by rotation of the messages). The output of the first mix becomes the input of the second mix, which in turn provides the input to the last mix. The output of the last mix is not linkable to the input of the first mix, nor to any of the intermediate stages.

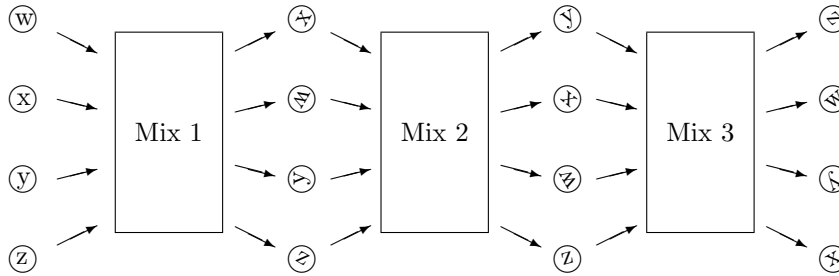


Figure 3.3: An example of a mixnet in operation (adapted from [HS00])

To prove that the output is indeed a permutation of the input, the mix can use zero-knowledge proofs or designated verifier proofs. A pair of mixes has another option available, called randomised partial checking [JJR02]: they shuffle and publish their results. An independent party then selects for each message in the intermediate set (i.e. the output of the first mix, which is the input of the second mix) whether the link to the input or the link to the final output is revealed. As the mixes do not know in advance which link they must reveal, the likelihood of being caught should they mix fraudulently is substantial. At the same time, input messages cannot be linked to the output messages as only one of the two links necessary for linking is revealed for any message.

Application in voting Mixnets are used to provide anonymity in various ways in voting systems. One approach is to use mixnets to construct an anonymous channel (a communication channel that hides who put a message on the channel). Another use is to shuffle the list of candidates on a specific ballot. The order of the candidate is then revealed to the voter (using a designated verifier proof) and correctness of the shuffle is proven publicly (using zero knowledge proofs). A third option, proposed in [Nef01], is to mix voter credentials together, such that the voters are no longer individually identifiable, but only as a group.

3.2 Voting schemes and systems

Academic interest in development of voting systems was triggered by a remark in a paper by Chaum on mix networks [Cha81]. The paper discussed the application of mix networks to voting in one short paragraph. In the years following Chaum's remark, quite a number of *theoretical* voting schemes were proposed. In order to ensure voter trust in the scheme, most of them offer traceability of the vote. Upon voting, the voter acquires a receipt, which she can use to trace her vote to the results. Benaloh and Tuinstra showed in [BT94] that such receipts can be used to nullify vote-privacy, when the voter shows her receipt to another party. They had the insight that vote-privacy is not optional, but must be enforced by the system. None of the classical schemes that had been proposed up to then satisfied this requirement. Their work led to a new direction in voting schemes: receipt-free voting systems. Later, Juels, Catalano and Jakobsson [JCJ05] further refined the notion of privacy by identifying the need for coercion-resistance (see the upper band in Figure 3.4).

Ensuring correctness of receipt-free and coercion-resistant systems is done by complex cryptographic operations. This convoluted approach to verification hindered their adoption in real-world elections. Recognising this as a problem, Chaum [Cha04] proposed a voting scheme which offered a far easier mechanism to verify correctness of the result. This inspired further developments of *practical* systems that combine verifiability and privacy. This particular type of systems is called end-to-end verifiable systems (lower band in Figure 3.4).

The aim of this section is to give an overview of the significant trends in voting, both theoretical schemes and practical systems. To this end, we discuss a number of significant instances of both of the above sketched types (schemes and systems). The discussions below focus on properties of and relations between the schemes and systems. For full specifics on the schemes and systems, the reader is referred to the original papers.

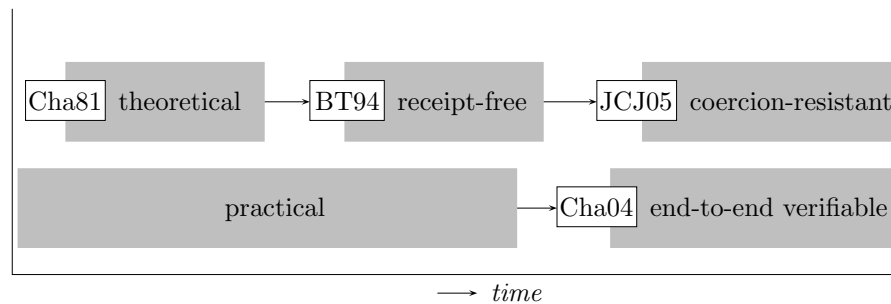


Figure 3.4: Development of voting systems

The rest of this section is organised as follows. First, theoretical schemes are described (following the upper time line in Figure 3.4). Then, practical systems are described (the lower time line in Figure 3.4).

3.2.1 Receipt-free voting schemes

There is a vast body of literature describing voting schemes predating (or not accounting for) the seminal work of Benaloh and Tuinstra. These schemes strove to provide privacy while assuring the correctness of the result. Below we highlight two such schemes: the FOO92 scheme [FOO92] and the CGS97 scheme [CGS97]. The FOO92 scheme was designed with the intent of actual use. It has been used as the basis for implementation of two voting systems (Sensus [CC97] and Evox [Her97]). CGS97 was designed with the intent of being as efficient as possible. As such, it has been used as a basis by various theoretical schemes that aim to incorporate receipt-freeness.

After these two schemes, the focus turns towards receipt-free schemes. First, we discuss the seminal work by Benaloh and Tuinstra [BT94]. This is followed by an alternative proposal by Niemi and Renvall [NR94], which appeared around the same time and addressed the same concerns. From then on, research into voting became quite prolific, and we highlight the more influential receipt-free schemes that followed their work.

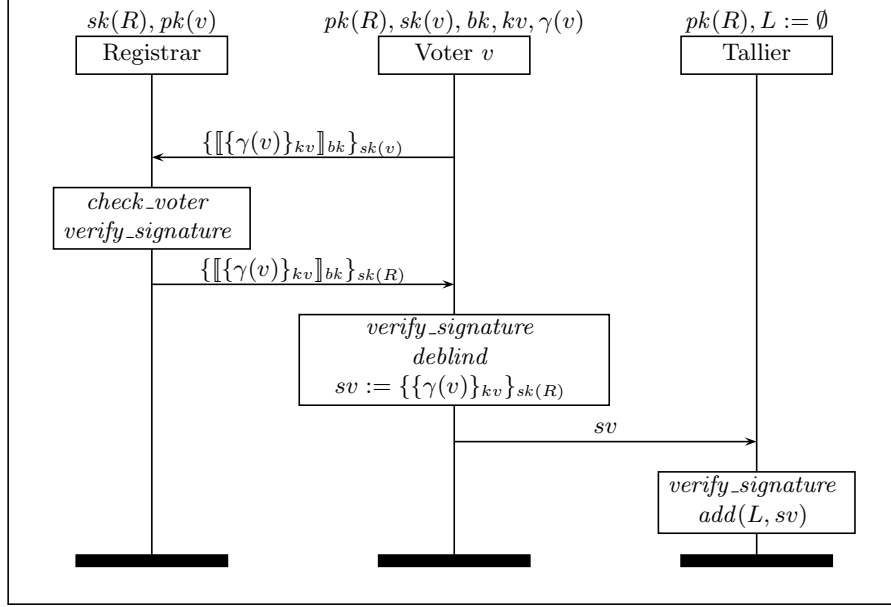
After discussing receipt-freeness, we outline the recent work of Juels, Catalano and Jakobsson [JCJ05]. Their introduction of the notion of coercion-resistance has led to a new direction in voting. However, this notion has not fully matured, and it is not clear yet which of the proposed coercion-resistant schemes will be regarded as influential. Hence, we cease our survey into receipt-free voting schemes at that point, and summarise the survey with two tables listing the claims and used cryptographic techniques, respectively.

FOO92

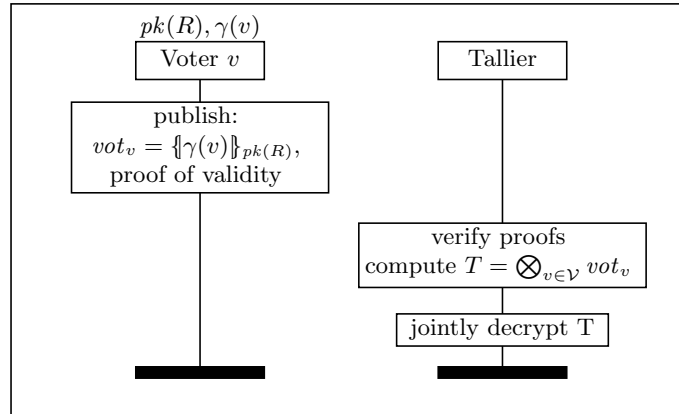
The FOO92 scheme [FOO92] aims to combine privacy with verifiability. To achieve this, it uses blind signatures as follows (see Figure 3.5). The voter encrypts her vote, blinds that encryption and sends the blinded, encrypted vote to the registrar (the first message of Figure 3.5). The registrar verifies that the voter is an eligible voter and that she hasn't voted yet. If so, the registrar signs the blinded, encrypted vote and sends the signed version to the voter (the second message in Figure 3.5). The voter then unblinds this message and obtains an encrypted vote, signed by the Registrar. She sends this via an anonymous channel to the tallier (the final message in Figure 3.5).

Then, after the voting period has ended, the tallier publishes list L containing all signed, encrypted votes received. After publication, the voter sends the decryption key for her vote to the tallier (not shown in Figure 3.5). This allows the tallier to open the vote and count the votes.

The FOO92 scheme has some known issues, for example all voters must take action twice, once to cast their votes and once to send the used encryption key. Furthermore, it is trivially not receipt-free: the decryption key of a particular vote is a sufficient receipt. The privacy of the FOO92 scheme is examined in detail in Chapter 5.

Figure 3.5: Vote casting in *FOO92***CGS97**

The scheme proposed by Cramer, Gennaro and Schoenmakers [CGS97] aims to be optimally efficient for a multi-authority setting. In their scheme, a voter publishes a single encrypted vote plus a proof that the published message contains a valid vote. The scheme is a homomorphic (ElGamal) voting scheme, it sums all encrypted votes and then uses joint decryption to decrypt the sum. Cramer et al. note that the proof of validity used can be made non-interactive using the Fiat-Shamir technique [FS86]. They note that in this case, care should then be taken to ensure that the constructed challenges are voter-specific. The scheme is sketched in Figure 3.6.

Figure 3.6: Vote casting in *CGS97*

Cramer et al. note that their scheme is not receipt-free. They state that this can be achieved using special channels, which in their view could be replaced with tamper-resistant hardware. They raise the idea of receipt-freeness without untappable channels, similar to incoercible multi-party computation without untappable channels. However, [HS00] points out that in the specific multi-party example cited, participants can prove specific behaviour.

The CGS97 scheme successfully reduced the communication and computational load for voting significantly in comparison to previous efforts. In addition, the ideas set forth by this scheme have been influential, leading to various adaptations to achieve receipt-freeness.

BT94

In their landmark paper, Benaloh and Tuinstra describe an attack on vote-privacy, apparently used in Italy. In certain election schemes used for multiple parallel elections in Italy, voters could list their votes in any order. Knowing this, an attacker can assign specific permutations of votes a priori. This permutation then acts as a voter's "signature". If a voter's signature is not in the set of cast votes, the voter deviated from her assigned voting strategy. If the signature is present, then the voter complied with her assigned voting strategy.

The above example illustrates two important things. First, it exemplifies that if the voter can somehow undo the privacy of her vote, she is at risk of being coerced. Secondly, it also shows that *any* input of the voter may allow her to lift her privacy.

Benaloh and Tuinstra devised a way around this problem. Their solution uses some cryptographic tricks and some strong assumptions to ensure privacy:

- A doubly homomorphic randomised encryption system, having ciphertext operations \otimes and \oslash with plaintext equivalent operations $+$ and $-$, respectively. The cryptographic system allows one to prove that a given ciphertext encrypts a given plaintext.
- A source of random bits.
- A voting booth.
While inside a voting booth, a voter cannot be observed, nor can the voter send out communications. However, the voter can still receive communications, and can also record received communications.

Moreover, there are only two candidates: 0 and 1.

The idea behind the main part of the scheme is that the voting authority first commits to a series of encrypted pairs of 0 and 1 privately sent to the voter. Then, the voting authority proves publicly that all but the first generated pair either consist of an encrypted 0 and an encrypted 1, or that they encrypt the same pair (possibly in reverse order) as the first pair. Which option is chosen depends on random bits generated by the source. As the voting authority committed to the decryptions of all pairs beforehand (including the first pair), the voter knows the ordering of all pairs and thus knows which value encrypts her vote.

Note that in addition to a specifically tailored cryptographic system, the BT94 scheme relies on the existence of a voting booth, and a private channel. While a cryptographic system satisfying the particular requirements exists, Benaloh and Tuinstra leave it unclear how to satisfy these latter two assumptions. In addition, their scheme can only handle a maximum of two candidates. Ensuing approaches sought to address some of these shortcomings, as is described further on.

NR94

Around the same time, Niemi and Renvall remarked the possibility for vote buying as well [NR94]. They too noted that all existing schemes gave the voter a token, which is made public upon counting. Their solution was to adapt this token, so that the voter cannot prove that it is hers. The (multi-party) computations done for this occur inside a voting booth, which keeps the voter completely private. The token allows a voter to check if her vote is part of the public set of received votes (ensuring individual verifiability). Furthermore, the correctness of the count can be publicly verified (ensuring universal verifiability).

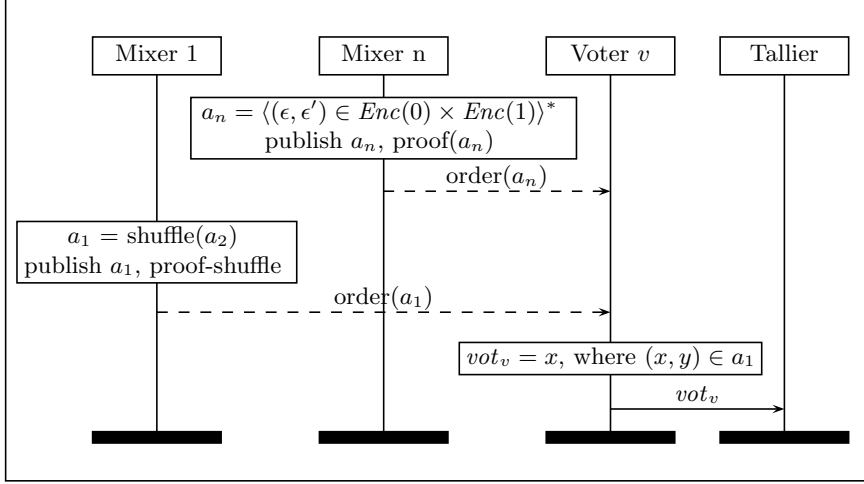
However, the ability for a voter to check that her vote is in the published set of votes can be abused by a voter to prove how she voted. Simply put, a voter can predict a specific vote occurring in the set (i.e., her own vote). She does so to the vote buyer, who then verifies that the value indeed appeared. In this way, the voter can prove to the vote buyer which vote is hers.

The computations needed do not scale well with respect to false votes, something which Niemi and Renvall already note. Furthermore, the scheme requires a strict assumption (a voting booth). Although their proposal was influential for its ideas (like [BT94]), the mechanics they proposed have been relegated to the fringes of voting research.

SK95

Sako and Kilian [SK95] noticed the reliance on strong physical assumptions (voting booths) in both [BT94] and [NR94]. They propose to use mixnets to relax the strong assumptions. Moreover, their mixnet-based approach is one of the first to use proofs of correctness for mixnets to achieve universal verifiability. Similar to [BT94], their scheme lets the voting authorities mix an encrypted 0 and an encrypted 1 vote, while proving to the voter how they mixed. This process is detailed in Figure 3.7.

The last mixer, mixer n in Figure 3.7, generates a list of encrypted zeroes and ones, using a probabilistic encryption scheme (ensuring no two ciphertexts are equal). The mixer publishes the list of encrypted pairs, and a proof that each pair contains an encrypted zero and an encrypted one. Furthermore, the mixer sends a proof to the voter of the order of each pair. This proof, sent over an untappable channel, is such that the voter can lie about the order. After this, the next mixer ($n - 1$, omitted in the depiction) takes a_n and shuffles it. Mixer $n - 1$ publishes the mixed list a_{n-1} , and a proof that he shuffled correctly. Moreover, he sends a proof of how he reordered over an untappable channel to the voter. Again, this proof can be used by the voter to lie. This process is repeated by all mixers in the mixnet. After the last mix, the

Figure 3.7: Voting in *SK95*

voter sends one element of one pair of a_1 (that corresponds to her choice) over an anonymous channel to the tallier.

The proposed mixing scheme was criticised by Michels and Horster [MH96]. They describe a known attack whereby a malicious voter can relate her vote to an honest voter's vote, by merely using the encrypted version of the honest vote. This relation survives decryption. Thus, the dishonest voter can later test all published votes to see which two satisfy the relation, thus revealing (to the malicious voter) how the honest voter voted. In addition to this attack, Michels and Horster also describe a new attack against the mixing process. If all subsequent mixing centres collaborate, they can reverse the mixing by the immediate preceding mix. Note that this means that if there is only one honest mix (which should be enough to provide an honest mixnet), its shuffling can be completely negated by the rest of the mixnet.

Sako and Kilian manage to lessen the strict assumptions of previous schemes (absence of a voting booth). Instead, the scheme requires an anonymous channel and a private channel. However, their approach does not scale well to multi-candidate elections and their cryptographic approach does not offer full security. The scheme has been used as basis for an implementation, DigiShuff [FMM⁺02]. DigiShuff mitigates the critique to a certain extent. DigiShuff is discussed below.

Okamoto97

Okamoto [Oka97] aims to reduce communication and computation overhead by using blind signatures and an anonymous channel. He identifies a security flaw with respect to receipt-freeness in an earlier version of his work [Oka96] and proposes three new versions. In this work, Okamoto considers a voting scheme to be receipt-free if the voter can cast a vote other than the vote buyer's preferred vote. The set of received, encrypted votes is made public. Okamoto avoids the problem of [NR94] by allowing

each received vote to be opened in many ways.

All new versions have strong assumptions. Each uses an anonymous channel from the voter to the authorities. Additionally, the first and second schemes require untappable channels, while the third scheme uses a voting booth.

HS00

Hirt and Sako [HS00] set out to further reduce overhead and enable less strict physical assumptions. They cast a critical eye on previous work, showing a receipt in the multi-authority version of [BT94], noting the processing load of [SK95] and identifying the use of anonymous, secret communication channels in [Oka97] as prohibitive towards adaption. Furthermore, they note that the designated verifier proofs of [SK95] are only designated, if the verifier possesses the corresponding private key (in normal cases, this is assumed to be guaranteed by the key infrastructure). The proof is transferable if the verifier can prove she does not possess the private key.

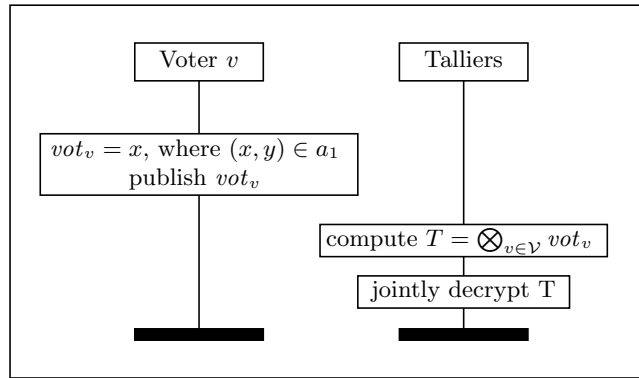


Figure 3.8: Change in vote casting from *SK95* to *HS00*

Hirt and Sako propose a generic construction for introducing receipt-freeness in voting schemes based on homomorphic cryptography. Much like [BT94, SK95], the scheme proposed by Hirt and Sako mixes a list of candidates. A zero knowledge proof of correctness of the mix is published, and a designated verifier proof revealing how the candidate list was mixed is given to the voter. The voter derives which entry in a shuffled, encrypted list represents her choice. This process greatly resembles vote casting in [SK95] (Figure 3.7). The only exception is the last message. Instead of sending her vote over an anonymous channel, the voter announces it publicly. The talliers take all published votes, sum them together and jointly decrypt the result, without reconstructing the decryption key. The publishing of the vote is depicted in Figure 3.8.

Like [SK95], the used designated verifier proofs require possession of a private key. Hirt and Sako provide a protocol in which a verifier proves she possesses the private key which ensures non-transferability.

Hirt and Sako apply their construction to [CGS97]. The lack of receipt-freeness

in [CGS97] originates from the fact that the voter creates and posts her vote on a public channel. This derivative consists of the basic construction by Hirt and Sako, and borrows the cryptographic scheme and the encoding of votes from [CGS97]. Consequently, the resulting scheme more closely resembles [SK95] (as depicted in Figure 3.7) than [CGS97] (as shown in Figure 3.6).

Hirt and Sako reduce the strong assumptions from earlier works. However, their approach still requires an untappable channel from voting authorities to voters. They conjecture that physical assumptions are necessary to ensure receipt-freeness – without such assumptions, the voter’s private information would be a valid receipt.

LK00

Lee and Kim [LK00] also set out to introduce receipt-freeness in [CGS97]. Unlike [HS00], Lee and Kim do not aim to provide a generic construction for receipt-freeness, but seek to incorporate receipt-freeness into [CGS97]. Their approach is to ensure that the receipt present in [CGS97] (the voter’s vote) can no longer act as a receipt. They introduce a trusted third party, an honest verifier (HV), that cooperates with the voter in constructing the vote.

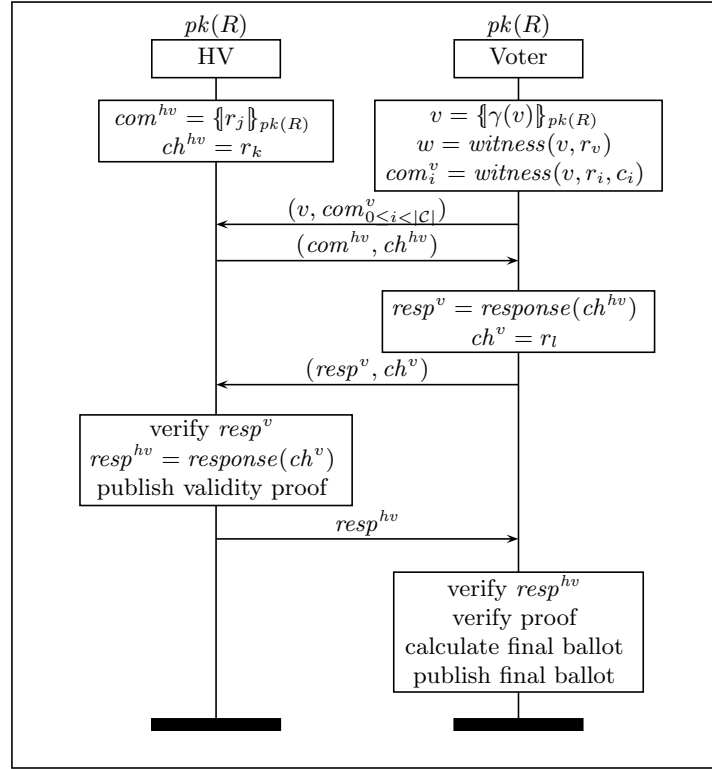


Figure 3.9: Extensions to *CGS97* for receipt-freeness in *LK00*

The vote encryption process (sketched in Figure 3.9) involves multiple steps. First, the voter encrypts her vote as v and commits to a witness to this vote, w . Furthermore,

she creates a false commitments $com^v(v, r_i, c_i)$ for every other candidate $c_i \in \mathcal{C} \setminus \{\gamma(v)\}$ and random numbers r_i . The voter's vote and all commitments are sent to HV. HV replies with its own commitment and challenge (which is independent of the voter's supplied information). The voter responds, and issues a challenge of her own. HV responds to this challenge and publishes a proof of validity for the final vote. The voter computes her final vote based on the HV's response, and publishes her vote.

The idea of the scheme is that as the voter no longer fully controls the encryption of her vote, one particular receipt is prevented. However, as Hirt points out in [Hir01], the HV can help the voter in casting a vote that falsifies the result. Furthermore, the voter can construct a receipt similarly to the receipt in the multi-authority version of [BT94].

The nature of the communication channel with the HV is not specified. However, the voter sends her vote unencrypted over this channel. Hence, it should be at least private.

BFP01

Baudron, Fouque, Pointcheval, Stern and Poupard seek to address the practical usefulness of voting schemes. To this end, they propose a scheme [BFP⁺01] that allows for tally computations at local, regional and national level. The scheme achieves this by having the voter cast three encrypted votes (one for each level) along with a proof that the three encrypted votes encrypt the same candidate.

As these votes are made public, the basic scheme is inherently not receipt-free. Baudron et al. address this by suggesting to use a hardware device to re-encrypt the votes. Correctness of the encryption can be proven using an interactive zero knowledge proof or a non-interactive, designated verifier proof. They claim that the interactive zero knowledge proof would not harm receipt-freeness as the actual transcript of the interaction is indistinguishable from a simulated transcript (and thus the voter can provide a fake transcript). However, as pointed out by Hirt [Hir01] and later by Juels et al. [JCJ05], the interactivity in a zero knowledge protocol can be abused to construct a receipt. Moreover, the remark in [HS00] concerning designated verifier proofs is also not addressed. As Baudron et al. do not provide further details, we consider the claim of receipt-freeness for their protocol dubious at best.

MBC01

Magkos, Burmester and Chrissikopoulos also set out to transform [CGS97] into a receipt-free voting scheme. They follow the approach of [LK00] by trying to ensure that the obvious receipt of [CGS97] is not a proof of how the voter voted. Again, the approach is to achieve this by having an external party co-construct the voter's vote. However, instead of relying on an external party, they propose to use a tamper-resistant smart card (see Figure 3.10). The voter sends a random order of the two possible votes (+1 and -1) to the smart card. The smart card then re-encrypts these values (denoted by *renc* in the figure) and sends them back. Finally, the smart card proves the correctness of the re-encryption using an interactive zero knowledge proof. After this, the voter knows which of a', b' is an encryption of her choice. After this,

the scheme mimics the steps of the CGS97 scheme.

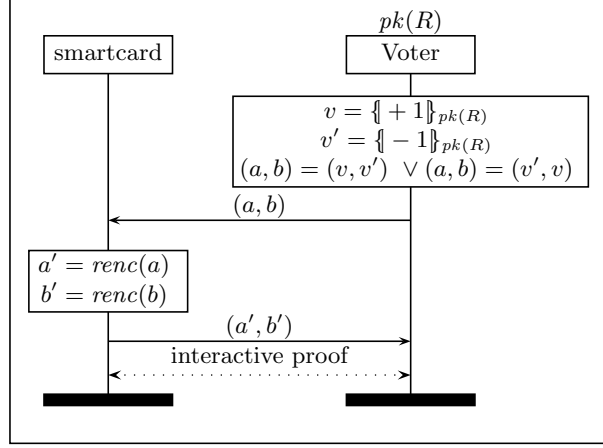


Figure 3.10: Extensions to *CGS97* for receipt-freeness in *MBC01*

Magkos et al. identify a need for randomness in voting schemes, to ensure vote-privacy. They analyse where this randomness could originate from: voter, authorities or an external source. According to their analysis, letting the voter introduce randomness enables her to abuse the randomness as a receipt. Letting the authorities choose the randomness implies a need for an untappable channel, according to this analysis. An external source would also require such a channel. However, Magkos et al. propose to provide the voter with tamper-resistant smart cards for introducing randomness. This would reduce the assumptions further: as long as the voter can interact with the smart card in an unobserved, untappable way, receipt-freeness can be achieved.

The approach of Magkos et al. somewhat relaxes the constraints on communication channels, but leaves open how to ensure that the voter is not observed while using these channels. Furthermore, their relaxation is done by introduction of a smart card. They do not answer the question whether this introduction brings about new risks.

KY02

Kiayias and Yung propose a scheme which takes quite a different stance [KY02]. They extend the notion of privacy with *perfect ballot secrecy*: knowledge of a result of a subset of ballots is only accessible to the coalition of all voters whose ballot is not in the subset. Then they propose a scheme that satisfies this criterion (which is weaker than receipt-freeness). Furthermore, they focus on the setting of boardroom elections.

In this scheme, voters provide randomness, then protect their vote using some randomness from all voters. This protection resembles homomorphic encryption, however, there is no decryption algorithm available. As with homomorphic encryption, the protected votes are added together, producing a protected result. To determine the result, the protection is removed by brute-force search.

The scheme requires voter-interaction at three different stages. The authors provide

mitigation procedures to ensure that a malicious voter cannot upset the entire election by abstaining in a later stage.

Although the authors mention receipt-freeness as a desirable requirement of voting schemes, they make no claim that their scheme is receipt-free, nor that receipt-freeness is or is not necessary in boardroom elections. The protection of the vote provides an obvious receipt. Hence, the scheme does not satisfy receipt-freeness. Interestingly enough, the scheme only uses a public channel, no specialised communication channel is used. This corroborates the remark in [HS00] that physical assumptions seem to be necessary to achieve receipt-freeness.

LK02

Lee and Kim take heed of Hirt's remarks [Hir01] and note that the approach to constructing a receipt for [LK00] is also applicable to [MBC01], as both voting schemes use interactive proofs, where the voter may abuse her interactivity to construct a receipt. They also claim that this holds true in general for blind signatures: the used blinding factor can act as a receipt.

Lee and Kim base their new receipt-free homomorphic scheme on the scheme proposed in [Hir01], but they replace the assumption of a trusted third party randomiser of [Hir01] with an assumption of a tamper-resistant randomiser possessed by the voter. Lee and Kim do note that the costs of such external hardware for every voter may be prohibitive to large-scale adaption. Although Lee and Kim do not state this explicitly, their scheme also relies on the assumption on designated verifier proofs, as pointed out in [HS00] (i.e., designated verifier proofs are transferable to anyone who can be convinced that the voter does not possess the decryption key). Furthermore, as Lee and Kim themselves point out, one of the proof-techniques they use may leak more information than intended.

In addition to the above remarks, the conclusions for [MBC01] carry over to [LK02]: the constraints are relaxed by introducing a new object, but the security of this new object is not extensively treated.

LBD03

Lee, Boyd, Dawson, Kim, Yang and Yoo [LBD⁺03] take the ideas of using tamper-resistant hardware from [LK02] and use it to construct a receipt-free mixnet-based voting scheme. The link between a voter and her vote is hidden by mixing. As such, the cast votes are individually decrypted to compute the tally. The scheme does not use the homomorphic property, even though the used cryptographic system, ElGamal, does support this. Verifiability of the result is achieved by verifiability of the mixnet and proofs of correctness from the tamper-resistant hardware.

Communication with the special hardware is, as in [MBC01] and previous works, assumed to be unobservable. Additionally, the scheme uses designated verifier proofs to prove correctness of mixing, which have the same assumption on knowledge of the secret key as [SK95, HS00, LK02]. Furthermore, Lee et al. note themselves that the malleability of the used cryptographic system enables a voter to copy another voter's encrypted vote. They seem to imply that this does not pose a security risk, even

though the signed and encrypted votes of each voter is cast by making it public.

ALBD04

Together with Aditya, Boyd and Dawson, Lee extends [LBD⁺03] to do away with the tamper-resistant hardware (by having that role fulfilled by a voting authority) and use an optimistic mixnet. The optimistic mixnet used is optimistic in the sense that it proves that the product of the input is the product of the output (and not that each element in the input corresponds to an element in the output). Note that this leaves the possibility for a dishonest mix to change one vote, as long as there is another change that cancels the first change (e.g., halving vote c_1 and doubling vote c_2 , as $\frac{1}{2}c_1 \cdot 2c_2 = c_1 \cdot c_2$). The optimistic mixnet on which Aditya et al. base their scheme pairs each input element c with its hash $h(c)$, and mixes the pair $(c, h(c))$. To further ensure that a dishonest mix does not thwart the entire mixing procedure, messages are doubly encrypted. In case of an (detected) error, the set of messages prior to the faulty mix is decrypted once. The decrypted set can then be used as input to a regular mixnet. In this way, the scheme ensures verifiability, while the mixnet provides receipt-freeness.

As Aditya et al. point out, the security of optimistic mixnets is not beyond doubt yet (see e.g. [Wik03]). They introduce several changes to the optimistic mixing scheme to prevent identified flaws. Most notable of these changes are the use of only one encryption layer, and omitting the hashing. This latter change reintroduces the possibility to change votes as explained above.

JCJ05

Juels, Catalano and Jakobsson noted in [JCJ05] that even if a scheme prevents vote-buying, there remain privacy-related threats for voters. They identify the following three privacy-related threats that are not covered by any privacy requirement on voting yet:

- Forced abstention (the voter is forced to abstain).
- Simulation (the voter is forced to give her credentials to the attacker, who votes in her stead).
- Randomised voting (the voter is forced to vote for a random candidate).

Juels et al. introduce the term *coercion-resistant* to describe the requirement of being receipt-free and preventing the three attacks above. The voting scheme they propose seeks to achieve this by allowing a voter to derive fake voting credentials from real voting credentials. The voter can claim the fake credentials are her real credentials. These fake credentials can then be used as the attacker pleases (either not, to abstain; or allowing the attacker to vote, or to cast a random vote). Votes cast using these fake credentials are removed at the tallying stage, after being made public and being mixed. As the false votes are removed *after* having been mixed, the link between a removed vote and a published vote is not retrievable.

System	Privacy			Verifiability		Channels	
	VP	RF	CR	Individual	Universal	$v \rightarrow a$	$a \rightarrow v$
[FOO92]	+	-	-	+	+	A	-
[CGS97]	+	-	-	+	+	- ¹	-
[BT94]	+	+ ²	-	+	-	A	U
[NR94]	+	+	-	+	+	A	U
[SK95]	+	+ ⁴	-	+	+	A	U
[Oka97]	+	+	+	+	+	A,UA	U
[HS00] ^{*,**}	+	+	-	+	+	-	U
[LK00] [*]	+	+ ²	-	+	+	P	P
[MBC01] [*]	+	+ ²	-	+	+	H	H
[BFP ⁺ 01]	+	- ³	-	+	+	H	H
[KY02]	+	-	-	+	+	-	-
[LK02]	+	+	-	+	+	H	H
[LBD ⁺ 03]	+	+	-	+	+	H	H
[ALBD04]	+	+	-	+	+	U	U
[JCJ05]	+	+ ⁴	+	+	+	A	U

* Improves [CGS97]

** Based on [SK95]

- Privacy claims: VP: vote privacy, RF: receipt-freeness, CR: coercion-resistance
- Channels: -: public, A: anonymous channel, P: private channel, U: untappable channel, UA: untappable anonymous channel, H: voter has hardware device

Notes:

1. The authors propose an extension that uses a private channel from voter to authority.
2. Claim known to be broken.
3. The authors propose an extension using standard techniques to satisfy receipt-freeness. Note that some of these techniques have been exploited to construct receipts in other schemes.
4. See the remarks in the discussion of the work.

Table 3.1: Claims of classical and receipt-free schemes

The authors claim that the published votes do not constitute receipts. Their reasoning is that the attacker does not know whether or not a voter has cast a vote, so the attacker cannot force a voter to decrypt any message. However, there is an effect on voter privacy, as follows. If a voter is willing to reduce her privacy, she can show how she constructed the encrypted vote. This proves the voter cast this one vote. This vote, however, may be false. But, if no votes are removed after mixing, then the vote was not false, and thus the voter has proven how she voted. In general, if the number of removed votes is smaller than the number of receipts obtained (by the attacker), voter privacy is harmed.

In short, Juels et al. further refine the notion of privacy in voting. They propose a voting scheme to satisfy this refined notion, which again relies on specific assumptions on the used communication channels. Furthermore, while their scheme allows a willing voter to reduce her privacy (i.e. the scheme is not receipt-free), that receipt does not prevent an unwilling voter to remain private. This indicates that coercion-resistance is not a straightforwardly stronger requirement than receipt-freeness.

	BlindSig	HomEnc	Mixnet	SecShare	ZKP	DVP
[FOO92]	+	-	- ¹	-	-	-
[CGS97]	-	+	-	+	+	-
[BT94]	-	+	-	-	-	+
[NR94]	-	-	-	+	+	+
[SK95]	-	-	+	-	+	+
[Oka97]	+	-	+	- ²	+	-
[HS00]	-	+	+	+	+	+
[LK00]	-	+	-	+	+	-
[MBC01]	-	+	-	+	+	+
[BFP ⁺ 01]	-	+	-	+	+	-
[KY02]	-	- ³	-	+	+	-
[LK02]	-	+	-	+	+	+
[LBD ⁺ 03]	-	-	+	+	- ⁴	+
[ALBD04]	-	-	+	+	+	+
[JCJ05]	-	+	+	+	+	+

Legend:

- BlindSig: blind signatures, HomEnc: homomorphic encryption, SecShare: secret sharing
 - ZKP: zero knowledge proof, DVP: designated verifier proof

Notes:

1. The use of mixnets is not described in the paper, but the scheme does use an anonymous channel (which are usually implemented using mixnets).
2. The second scheme in [Oka97] does use secret sharing, the other two presented schemes do not.
3. The result is determined by combining the protected votes, and deprotecting the result, much like homomorphic cryptography.
4. The use of zero knowledge proofs is not described in the paper, however, the paper suggests to use a verifiable mixnet (which would provide a zero knowledge proof).
5. This is only used if the registering authority is not trusted.

Table 3.2: Usage of cryptographic techniques in voting schemes

Summary

The claims made by the various schemes with respect to privacy and verifiability are listed in Table 3.1. The use of cryptographic techniques is summarised in Table 3.2 (note that re-encryption is used in mixnets, and, as such, has been left out of the table).

From Table 3.1, it is obvious that every protocol that claims receipt-freeness, uses a specialised communication channel. Hirt and Sako [HS00] remark this too, and hypothesise that some physical assumptions are necessary to achieve receipt-freeness. The following types of communication channels have been used in literature to ensure receipt-freeness.

- An anonymous channel, i.e. a channel where no one learns who sent a message.
- A private channel, i.e. a channel where no observer learns anything about the contents of messages.
- An untappable channel, i.e. a channel where no observer can learn anything, not even if messages are communicated.

- A hardware device, communications with which are untappable.
- An untappable anonymous channel, i.e. a channel which is both anonymous and untappable.

A frequently used concept in actual elections to ensure privacy is a voting booth. The concept of a voting booth is often realised in remote voting schemes as a bi-directional untappable channel. However, a voting booth is to ensure a private environment, while untappable channels ensure private communications. Despite these diverging objectives, an untappable channel can constitute a reasonable approximation, as it is a way to incorporate procedures and physical measures that are to ensure that a voter can communicate in private. Moreover, in actual elections the privacy of the voting booth is occasionally violated. For example, visually impaired voters may be allowed to bring a seeing assistant into the voting booth.

3.2.2 Practical and end-to-end verifiable systems

Elections were held before Chaum initiated research into voting. With the advent of computers and the Internet, governments and democratic bodies became interested in leveraging their potential for elections. On top of that, researchers have been on the lookout for a real-world use case for their schemes. Examples of this are given by the implementations of the FOO92 scheme, DigiShuff [FMM⁺02], and the RIES system [HJP05].

Chaum's aim to provide an unprecedented level of verifiability, whilst preserving privacy [Cha04], proved to be a fertile breeding ground for research. Following Chaum's work, various end-to-end verifiable systems have been proposed, all aiming to combine privacy and verifiability. In this section, we first discuss the implementations of the FOO92 scheme and of DigiShuff. This is followed by an overview of the RIES system. The rest of the section is dedicated to Chaum's work and the works it inspired.

Implementations of FOO92

There are two publicly available implementations of the FOO92 scheme [FOO92]: Sensus [CC97] and Evox [Her97].

Evox. Evox chooses to implement the anonymous channel by having a third party receive all votes. After the voting phase ends, the third party forwards the set of received messages to the tallier. This in itself is not detrimental to privacy. However, the Evox system deviates from [FOO92] by adding the decrypted vote to the message sent over the anonymous channel. This means that every voter reveals to the third party how she voted.

Sensus. Sensus does not provide an implementation of the anonymous channel. As such, it remains similar to FOO92 in this regard, thus avoiding the problem of Evox. However, the anonymous channel in FOO92 is a necessary constituent to achieve vote-privacy (with respect to the tallier). As Sensus does not provide an anonymous channel, it fails to provide vote-privacy.

Comparison with FOO92. Both implementations of FOO92 clarify one item left unclear in the original scheme. In FOO92, a list of all received encrypted votes is published. This list is used by voters to send their decryption key (indicating which index their vote has, so that the tallier knows which key to use). In FOO92, it is not clear whether or not the published list is updated with the keys. Both Sensus and Evox do update the published list. In this way, both systems ensure that anyone can construct any vote on the list. This addresses the problem of a voter that constructs a listed vote, thus proving she cast that vote.

While both Sensus and Evox base their design on [FOO92], both systems have deviations which lower the offered privacy with respect to [FOO92]. In the case of Evox, this can be mitigated by assuming a third party trusted to not reveal or act upon how voters voted. However, far easier systems are possible if a trusted third party is used.

DigiShuff

The DigiShuff system [FMM⁺02] is based on the work by Sako and Kilian [SK95]. Furukawa, Miyauchi, Mori, Obana and Sako notice that some of the computations involved for proving correctness of shuffling and proving correctness of decrypting are similar. By merging these computations, they achieve a speed-up in comparison to [SK95]. Furthermore, the system uses a “Shuffling Management Centre”, that handles communications with each mix in the mixnet. This prevents the attack of [MH96], as long as the Shuffling Management Centre does not cooperate with the individual mixes.

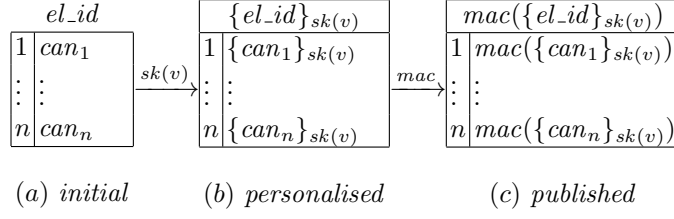
The DigiShuff system exploits the proposed merging of computations to great effect. Furukawa et al. claim speed improvements over [SK95] of up to a factor 16 in a test election with 10,000 voters.

RIES

The RIES system [HJP05] was developed in the Netherlands to support voting for water control boards (which are democratic bodies). The system aims to provide a high level of verifiability. Below, we describe how this is achieved.

The RIES system operates in three phases: pre-election, post-election and a voting phase. During the pre-election phase, the authorities generate an election id el_id and, for each voter, a secret key. This key is used to generate a personalised candidate list from the initial candidate list (see Figure 3.11). To generate this, each candidate is encrypted with the voter’s secret key. The election id is also encrypted with the voter’s key and acts as a pseudonym for the voter. This personalised candidate list is sent to the voter. Furthermore, a hash function, mac , is applied to each entry on the personalised list. This results in a new list (shown in Figure 3.11(c)), which is published. As the hash function is non-invertible, this does not provide information about the personalised candidate list.

To vote, a voter selects the candidate of her choice on her personalised candidate list. She sends her vote and her pseudonym over an anonymous channel to the tallier. When all votes have been received, the post-voting phase begins. The tallier publishes

Figure 3.11: Per-voter candidate lists in *RIES*

the set of received votes. To determine the final result, the *mac* of each received vote is computed. Using the supplied pseudonym, the resulting value can be looked up in the published lists and so counted for the correct candidate. The above procedures allow voters to verify that their vote is counted, and that it is counted correctly.

The security of RIES has been analysed by various academics (see e.g. [HJP05, JV07]) as well as by officially commissioned security institutions [HJS⁺08, GHH⁺08]. Although the parts examined (and thus the conclusions) vary from study to study, it is clear that the RIES design has various problems. The most serious of these are the inherent ability for a voter to prove how she voted, the possibility for the authorities to vote on behalf of voters, and the lack of procedures to deal with complaints (such as when a voter discovers that her vote is missing).

Chaum04

In [Cha04], Chaum introduced visual cryptography as a means to combine easy voter verification with vote-privacy. A voter's vote is recorded on two layers of paper. Each layer contains a pattern of apparently random dots, but when overlaying them over each other, the voter's vote is revealed. The voter can take the layer of her choice as a receipt, as it, by itself, does not reveal her vote. The other layer is destroyed. The receipt layer is scanned and acts as the voter's ballot.

All ballots are published, allowing the voter to verify that her ballot is received correctly. To determine the election result from the ballots, the ballots are mixed. Each mix visually decrypts the ballots. This results in a set of readable ballots, from which the tally is straightforwardly established. To ensure verifiability, Chaum suggests to use randomised partial checking [JJR02].

Chaum's approach to combining privacy and verifiability does not require heavy cryptographic operations on the part of the voter. However, mixing applied to visual cryptography makes the scheme less transparent.

Prêt à Voter

Prêt à Voter builds forth on the ideas proposed by Chaum. It was originally developed by Ryan [Rya04, Rya05], and later extended by Chaum, Ryan and Schneider [CRS05]. This scheme avoids the use of a two-layered receipt by having two columns. The left-hand column of the ballot presents the candidates in a random order, while the right-hand column has space for the voter to mark her preference. At the bottom of the

right-hand column is a cryptographic string (an onion) which encodes the candidate ordering on the ballot. This onion is the encryption of a random offset D_0 with each talliers public keys.

South	
North	
East	
West	
	3ei82Lk

Figure 3.12: A ballot in *Prêt à Voter*

A voter marks her preference, separates the two columns and destroys the left-hand column. The right-hand column is scanned by the system and taken home by the voter as a receipt. Since the candidate ordering is not retrievable from the right-hand column except by all talliers decrypting in order, the receipt cannot be used to prove the voter's preference.

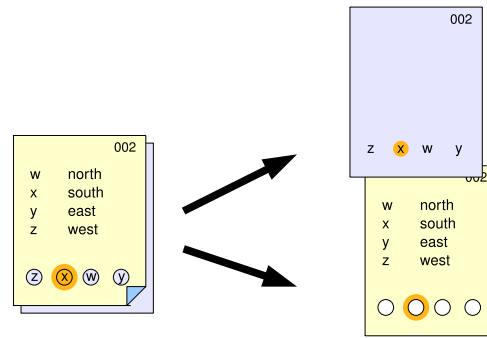
Creating a candidate order is handled in a similar manner to [HS00]. The main difference is that *Prêt à Voter* does not prove the candidate order. Instead, there are more ballots than necessary. A voter has the option of having the onion on an empty ballot decrypted, to confirm the candidate order of that ballot. In this fashion, *Prêt à Voter* enables a cut-and-choose style proof of correctness of the candidate order.

To provide further verifiability without sacrificing privacy, the conversion of receipts to countable votes is subject to randomised partial checking [JJR02]. This part of *Prêt à Voter* is lifted out by the subsequent voting system Scantegrity, which is detailed below.

In addition to these verification facilities, *Prêt à Voter* provides auditing procedures to ensure that the ordering on a given ballot matches that ballot's onion (voiding the ballot in the process, hence this procedure can only be executed on surplus ballots), and several error detection and recovery procedures.

Since its inception, *Prêt à Voter* has received further attention [RS06, LR08a], which addresses discovered problems such as chain voting and kleptography. Chain voting is a voting attack where an intruder somehow has a blank ballot. The intruder fills in the candidate of his choice on the ballot. Next, he offers a voter who is about to go and vote a reward for bringing back a blank ballot. As she must cast a vote, the intruder gives the voter his filled-in ballot. When the voter hands the intruder a blank ballot, he gives her a reward and once again possess a blank ballot. Kleptography is the trick of embedding a cryptographic system inside another cryptographic system. Users of the outer cryptographic system do not experience any difference, but their encrypted messages leak information to those possessing the decryption key of the inner cryptographic system. *Prêt à Voter* uses cryptography to hide the candidate ordering. As this is the key to achieving privacy, information leakage would potentially nullify privacy.

The extensions to *Prêt à Voter* have addressed these and other issues and have extended the capabilities of the system to support more diverse voting methods, leading to a more mature voting system.

Figure 3.13: A *Punchscan* ballot voting for “south”

Punchscan

Punchscan [FCS06] takes Chaum’s idea of layers quite literally. In Punchscan, a ballot consists of two layers (see Figure 3.13, where the layers are coloured for clarity). The top layer has a numbered list of the candidates in a random order. Below that list, there is a series of holes. The bottom layer provides the used numbers in a random order at the location of the holes in the top layer. A choice is made by using a big marker to colour both the bottom layer number and the top layer hole at the same time. The top layer does not identify which choice has been made, only which hole has been selected. The bottom layer does not reveal which candidate has been selected, only which sequence number that candidate had on this particular ballot. Hence, neither ballot can identify the voter’s choice. One layer is destroyed, and the other one is copied for tallying. The voter retains the copied layer for verification.

The ordering of sequence numbers in both top and bottom layer is stored by the authorities, and linked to the ballot id number (in Figure 3.13, the ballot has id number 002). Using the stored information, the authorities can reconstruct the voter’s choice from either layer in a manner similar to Prêt à Voter.

Although Punchscan seems to offer privacy and verifiability, the scheme has drawbacks. One such drawback, as mentioned in [BMR07], is that even though a layer does not give sufficient information to determine how a voter voted, the distribution of layers over possible votes can be skewed. Consider, for example, an election between two choices (Figure 3.14).

A: Yes B: No a b	A: Yes B: No b a	B: Yes A: No a b	B: Yes A: No b a
------------------------	------------------------	------------------------	------------------------

Figure 3.14: Ballot orderings for a two-choice election in Punchscan

Consider an attacker who offers a reward for:

- any top-layer where Yes is labelled A, and where the left hole is marked,
- any bottom layer where A appears left and is marked.

All possible ballots are shown in Figure 3.14. Note that only for the first ballot, voting Yes will leave the result with a rewarding layer. To receive a reward if given the second or third ballot, a voter must vote No. And, if given the fourth ballot, the voter cannot produce a layer that will receive the reward.

This particular drawback was addressed in Punchscan by forcing a voter to choose which layer is her receipt before she has seen the ballot.

Scantegrity

The researchers behind Punchscan recognised that the process of randomised partial checking and reconstructing the choice (originating from Prêt à Voter) can be taken independent from the ballot-casting process. They devised a new verification system called Scantegrity [CEC⁺08]. Scantegrity can be added to an existing voting system to provide verifiability. In essence, Scantegrity employs randomised partial checking [JJR02] as follows: it uses two mixes to mix the votes, making the results of each mix public. Verifiability of the entire mixing process is achieved by revealing for each vote in the intermediate mixed set either where it originated from (revealing the link to the input), or where it ended up (revealing the link to the output).

In Figure 3.15 this process is more detailed. Again, each choice on a ballot is assigned a random code $\in \{w, x, y, z\}$. The Switchboard is the result after the first mix (and thus the input to the second mix). The Switchboard is published. Verifiability is achieved as follows: for each entry on the Switchboard, a coin is flipped. Depending on the result of this coin flip, either it is revealed where it originated (indicated by thick lines in the left mix), or it is revealed how it affects the result (indicated by thick lines in the right mix).

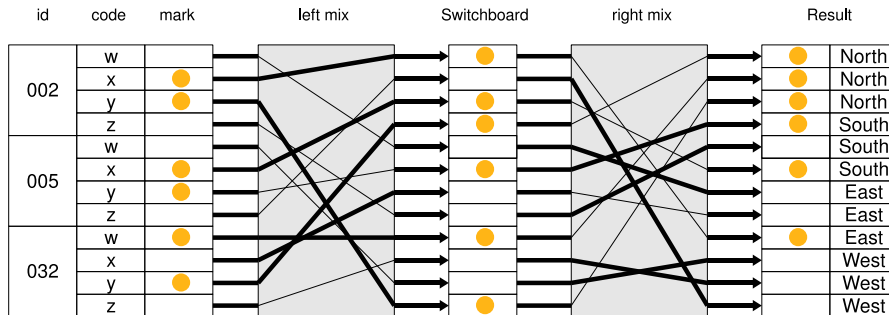


Figure 3.15: Vote verification in *Scantegrity*

Both Scantegrity and Punchscan suffer from security problems. The pattern voting attack (as described in [BT94]) is not prevented by either. Furthermore, they do not prevent randomisation attacks [JCJ05].

Due to detected security and usability problems, several researchers of the original Scantegrity teamed up with others to develop Scantegrity II [CCC⁺08]. The main improvement is hiding the verification codes using invisible ink. Using a special marker, the voter marks her choice, which reveals the invisible ink (and thus the confirmation code for that choice).

Scratch & Vote

Much like Prêt à Voter and Punchscan, Scratch & Vote [AR06], by Adida and Rivest, uses randomised candidate orderings on the ballots. A ballot lists the candidates in the left column and has space for the voter to indicate her choice in the right column. Below the right column, there is a bar code which states the candidate ordering in encrypted form. Below the bar code, there is a scratchable surface. When this surface is removed, the random factors used to encrypt the candidate ordering are revealed. To verify that the order in the left column matches the bar code in the right column, a voter can scratch away the scratch-layer and then verify the ordering of the bar code (using a bar code scanner).

For example, consider an election with options North, West, South and East (see Figure 3.16).

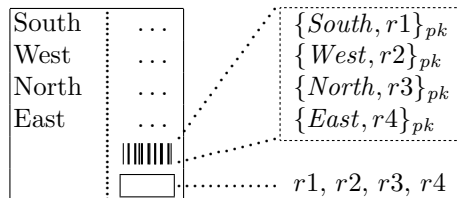


Figure 3.16: A ballot in *Scratch & Vote*

A voter receives two ballots, one to prove ballot integrity and one to vote. The voter chooses which ballot is used for which purpose. To prove ballot integrity, the voter removes the scratch surface, revealing the used randomisation values. Using a bar code scanner, the voter can now compute whether the bar code candidate ordering (which is encrypted) matches the depicted candidate ordering (which she can encrypt herself, using the revealed random values). The used ballot is invalid, as its scratch surface has been removed. Therefore, if the voter is satisfied, she uses the other ballot to cast her vote. Upon casting this ballot, its scratch surface is detached from the ballot and destroyed. In this way, the ballot no longer holds sufficient information by itself to reveal the decrypted candidate ordering.

The election result is established by applying the homomorphic property of the used cryptographic system. The encrypted result is then decrypted without revealing the decryption key (as described in Section 3.1.3).

Adida and Rivest extend the concept of scratch-surface verifiability to both Prêt à Voter and Punchscan. However, they do note that privacy of the scheme is based on the assumption that the scratch surface is perfect: either it hides the used random values, or it is detected as having been scratched. Furthermore, malicious authorities could record the candidate orderings in advance, and use this to reveal how voters voted. Adida and Rivest remark that this problem is inherent in preprinted paper-based cryptographic voting. For example, Prêt à Voter and Punchscan have the same weakness.

3BS

A radically different approach to end-to-end verifiability was proposed by Rivest and Smith. Their scheme, the ThreeBallot scheme or 3BS [RS07], uses a specific form of ballot. This ballot consists of three separate ballots, which together form the whole ballot (a Threeballot, see Figure 3.17). Each separate ballot contains an election-wide unique code at the bottom.

To vote, a voter places one mark in each row. In the row of her choice, she places a second mark. The voter is free to place the marks in any column, as long as there is one row with two marks (her choice) and all other rows have one mark.

When the voter casts the vote, she chooses one ballot of which she receives a certified copy. The copy allows the voter to verify that her Threeballot contributes to the result. After voting, the set of all received ballots is published. This allows all voters to verify that their certified copy is in the set, as well as verifying that the announced result matches the published set of ballots. The certified copy does not prove how the voter voted, as it could have been part of various valid Threeballots. The ballots in the published set can be combined with the certified copy of a ballot. Although not all possible combinations constitute a valid Threeballot, in general there will be several possible valid Threeballots, thus ensuring vote-privacy.

ballot 1a	ballot 1b	ballot 1c
North ○	North ●	North ○
South ●	South ○	South ○
East ●	East ○	East ●
West ○	West ○	West ●
\$xY63#bZ['@a0^U_3G<]Lw%4!r;}

Figure 3.17: A *Threeballot* in favour of “East”

As Rivest and Smith already note, not all possible ballots can be combined with a given receipt. For example, a receipt on which each candidate has a mark must be linked to a completely unmarked ballot and a ballot where precisely one candidate is marked. This means the privacy of a voter can be significantly reduced. The authors propose several solutions to address this. Each solution has its own problems with respect to feasibility and practicality.

Alongside 3BS, Rivest and Smith introduce two additional voting schemes in [RS07]: VAV (vote/antivote/vote) and Twin. VAV greatly resembles 3BS, but interprets the ballots differently. Two ballot are marked ‘V’ for vote, the third is marked ‘A’ for antivote. A mark on the antivote ballot cancels a mark on a vote ballot. The VAV scheme requires that the antivote ballot is marked precisely the same as one of the two vote ballots. Other than this, the scheme works as 3BS (e.g. the voter can choose which ballot to copy as a receipt). In the third proposed scheme, Twin, each voter receives a copy of a receipt of a vote, but not of her own vote. In this way, direct verifiability is traded off for increased privacy.

Rivest and Smith already point out several possible avenues of attack on 3BS in their original paper. The level of privacy 3BS offers has been the subject of criticism

(see e.g. [App06]). Nevertheless, 3BS's core ideas offer an interesting approach to combining verifiability and privacy. 3BS's approach is further explored in Section 5.3.

Bingo Voting

Bohli, Müller-Quade and Röhrich propose a radically different approach: Bingo Voting [BMR07]. In Bingo Voting, the authorities generate, for each voter, one random number per candidate, generating a pair of random number and candidate. For voter v_i and candidate c_j , such a pair would look like $(r_{i,j}, c_j)$. Thus, for n voters and m candidates, the authorities generate $n \cdot m$ pairs. The authorities keep these pairs secret, but publish commitments to these pairs.

To vote, a voter selects her candidate. This candidate is assigned a freshly generated random number. To ensure freshness of the generated number, Bohli et al. suggest to use (for example) a mechanical device such as used in bingo. The voter is given a receipt, on which the candidate of her choice is assigned the freshly generated random number. To ensure that the receipt cannot be used to sell her vote, all other candidates are assigned one of the previously generated random numbers.

In order to establish the election result, the list of all handed-out receipts is published. Furthermore, a list of all unused pairs is published and opened, and the authorities prove that all unopened commitments are indeed used on one receipt. The final result is then given by the unused, opened candidate-number pairs. Note that an abstaining voter leaves one unused candidate-number pair per candidate. Hence, the abstaining voters can be deducted from the tally, if necessary.

Bohli et al. claim that Bingo Voting satisfies (amongst others) Cast-as-Intended, because a voter can verify that the freshly generated number is assigned to the candidate of her choice. However, during casting the voter has no opportunity to verify that the other numbers assigned are numbers that were committed to previously, nor that exactly one number was used from each set of candidate-number pairs of the non-preferred candidates.

Aperio

The Aperio system [ECA08] is a variation on Punchscan that was developed to serve as an end-to-end verifiable system for environments where computerised support is low (e.g. developing countries). Aperio extends the layered approach of Punchscan further, basically providing a layer-based approach to randomised partial auditing. Each ballot has a receipt layer and two (differently coloured) audit layers. Each layer has its own unique identification number for auditing. Each layer is marked as the voter marks her choice on the top layer (the ballot), recording the location of the voter's choice. The election officials create, for each colour of audit layer, two sealed envelopes. The envelopes each contain a list, the one list linking the audit identification numbers to the voter receipt identification numbers. The other list links the audit identification number to the used candidate ordering.

Randomised partial auditing is achieved by choosing which colour of layer will reveal which of the two links. As both links (the link to the voter ballot and the link to the result) are revealed, the system provides a level of verifiability.

Conclusions

Various efforts have emerged that sought to transform theoretical voting schemes into actual systems. The systems created aim to combine strong privacy measures with a high level of verifiability. Chaum’s work [Cha04] inspired new ways to leverage the privacy and verifiability-combining properties of mixnets. However, care must still be taken – with the cryptographic details (e.g. the flaw in [SK95], corrected by DigiShuff) as well as with the new details (e.g. the privacy-reducing threats to Punchscan and 3BS).

3.3 Summary

A wide variety of theoretical “receipt-free” schemes has been proposed. Despite arguments (and even proofs) of achieving receipt-freeness, methods to construct a receipt for several of these schemes have emerged in literature.

A wide variety of practical “end-to-end verifiable” systems has been proposed. Despite arguments (and even proofs) of achieving privacy and verifiability, methods to reduce a voter’s privacy have been identified for several systems.

Claims of privacy are far easier made than substantiated. As established in the previous chapter, a better understanding of privacy in voting is necessary. But this is not enough. We need a rigorous, well-developed method to provide design methods and verification methodology that can actually *prove* privacy. Hence, privacy needs to be unambiguously defined, and a verification methodology needs to be designed that can verify whether a voting scheme or system fulfills the unambiguous definition of privacy. The area of formal methods offers a structured and rigorous mathematically based approach to definitions and verification.

In the next chapter, we design a formal framework for formalising voting schemes and systems, which supports precise articulations of privacy properties.

Formalising voting systems

From the previous chapters, a need for formalising privacy in voting has emerged. In this chapter, we investigate existing formalisations and propose a new, improved formalisation. The chapter is organised as follows.

First, we discuss previous formalisations of privacy in voting and outline our approach in Section 4.1. A formal framework for modelling voting systems is introduced in Section 4.2. Section 4.3 shows how to quantify privacy in the framework. Then, the attention turns towards conspiring voters. Section 4.4 discusses ways in which voters can conspire to reduce their privacy. These ways are formalised in Section 4.5. The chapter is ended by discussing the relation between privacy as captured by the framework and receipt-freeness, coercion-resistance and verifiability in Section 4.7.

4.1 Related work

This section discusses the various research efforts that have been devoted to formalising privacy properties for electronic voting. We can distinguish two main lines of research into formalising privacy in voting. On the one hand we see modelling of privacy properties in applied π by Kremer and Ryan [KR05], and later Kremer, Ryan and Delaune [DKR06, DKR08], which culminated in the automatic verification of Backes, Hrițcu and Maffei [BHM08]. On the other hand we see works formalising privacy in logics: Jonker and De Vink [JV06] in first-order logic, and then privacy in epistemic logic by Jonker and Pieters [JP06], and later Baskar, Ramanujam and Suresh [BRS07], who have a knowledge based formalisation in epistemic logic.

In addition to the lines sketched above, Juels, Catalano and Jakobsson introduce *coercion-resistance*. Their definition is of a cryptographic nature, in that it is a comparison of probabilities. The definition compares the probability of the intruder to determine if a coercion attempt was successful to an “ideal” intruder. The “ideal” intruder is one that does not learn whether or not coercion was successful.

In the remainder of this section, we first discuss the research formalising properties in applied π . A discussion of the logical formalisms follows. We finish by describing our approach to furthering the definitions of privacy.

4.1.1 Privacy expressed in applied π

Kremer and Ryan analyse the FOO92 protocol in the applied π calculus [KR05]. There they provide an early formalisation of privacy. This formalisation is based on the indistinguishability of swapping two votes between voters. Swapping votes between voters ensures that the result of the election remains unchanged, while having a different input to the system. Kremer and Ryan formalise indistinguishability as observational equivalence.

In their later work with Delaune [DKR06, DKR08], indistinguishability is defined as labelled bisimilarity instead, which coincides with observational equivalence. Delaune, Kremer and Ryan use labelled bisimilarity as the basis for vote-privacy, receipt-freeness and coercion-resistance. They propose that a system is receipt-free if an intruder cannot distinguish between a cooperating, information-sharing voter and a voter who fakes cooperation with the intruder. To ensure the result of the two cases are equivalent, there must be a second voter who swaps votes with the voter under observation. Delaune, Kremer and Ryan extend this approach to encompass coercion-resistance, which they view as the situation where the voter not merely shares information with the intruder, but in addition only sends out intruder-supplied terms in the process of casting her vote. The definitions are applied to the scheme proposed by Lee et al. [LBD⁺03]. To determine whether the scheme is receipt-free, a “faking voter” needs to be constructed, who fakes cooperation with the intruder. A similar construction holds for coercion-resistance, where Delaune, Kremer and Ryan note that the scheme by Lee et al. is not coercion-resistant if the encryption system used supports integrity checking.

Backes, Hritcu and Mattei [BHM08] extend the work of Delaune, Kremer and Ryan. They address several drawbacks: the definition of coercion-resistance in [DKR08] uses quantification over an infinite set of contexts, forces the intruder to follow the intended behaviour of the system, and does not address forced abstention attacks. Backes et al. propose a new definition of coercion-resistance, which captures immunity to simulation attacks and thus (as the authors show) forced abstention attacks, while leaving randomisation attacks as future work. Backes et al. define coercion-resistance as a comparison between a voter feigning cooperation with the intruder to a voter who is truly cooperating with the intruder. This definition requires two additional agents to prevent trivial differences in the result. There is a second voter to balance the vote of the cooperating voter, and an extractor that balances any vote the intruder may supply himself, using the voting credentials supplied by the coerced (or feigning) voter. In this way, the election result is the same for both scenarios (coerced voter vs. feigning voter). This definition is then applied to the work of Juels et al. [JCJ05].

4.1.2 Privacy expressed in logics

While the above line of research has been fruitful, it fails to acknowledge the role of the intruder knowledge. Intuitively, receipt-freeness is about proving something to the intruder. As such, the intruder’s knowledge comes into play. Jonker and De Vink [JV06] characterise receipt-freeness as these properties of the receipt. In their view, a receipt is an object that uniquely identifies the voter, that uniquely identifies the voter’s candidate and that uniquely identifies the voter’s cast vote. Hence, their

definition of receipt-freeness is based on the absence of such objects. Jonker and Pieters [JP06] build on these ideas to provide a definition of receipt-freeness based on the anonymity framework by Garcia, Hasuo, Pieters and Van Rossum [GHPR05]. They distinguish *weak* receipt-freeness from *strong* receipt-freeness. Weak receipt-freeness means that the intruder is not sure if a specific voter voted for a specific candidate. Strong receipt-freeness explicitly includes a set of candidates which the intruder cannot rule out as possible choices of the voter. Both of these works express receipt-freeness in a generic, logical formalism, which is not easily applied. Baskar, Ramanujam and Suresh [BRS07] continue in this line, focusing on the knowledge of agents. Their approach establish a more practically applicable decision procedure for receipt-freeness. They also prove that derivability of a term from a set modelling knowledge is decidable in polynomial time. Baskar et al. define receipt-freeness in an epistemic logic, much like Jonker and Pieters. While expressing privacy properties in a logic of knowledge is quite natural and straightforward, it is more difficult to develop verification techniques within such a framework.

4.1.3 Approach

The definitions of [DKR06, DKR08, BHM08] provide qualitative definitions of privacy properties. A qualitative approach leaves the possibility for the intruder to reduce privacy, however. Examples of such include forcing a voter *not* to vote for a certain candidate, and determining at which end of the political spectrum the voter voted.

We believe that the only way to detect such privacy attacks, is to compute the precise set of possible votes the voter may have cast. Consequently, we take the view that *any* modifier of privacy constitutes a receipt, as inspired by the definitions of weak and strong receipt-freeness of [JP06]. This implies that we do not express privacy as a comparison between two situations, but the privacy of a given situation is measured. To this end, the distinguishing power of the intruder needs to be captured precisely. This distinguishing power derives from his knowledge, for which Baskar et al. [BRS07] provide a knowledge modeling which is decidable. The distinguishing power of that knowledge is captured by Garcia et al. in [GHPR05], where they provide a definition of reinterpretation. To measure the exact privacy, we introduce anonymity groups, along the lines of Mauw et al. [MVV04] and Chothia et al. [COPT07].

4.2 Framework for modelling voting systems

In this section, we develop a formal syntax and provide semantics for expressing the communication behaviour of voting systems. We use the following setting for voting systems.

- There is a set of voters \mathcal{V} , a set of candidates \mathcal{C} , and a set of voting authorities Aut .
- Each voter $v \in \mathcal{V}$ is entitled to cast one vote for a candidate from the set of candidates \mathcal{C} .
- A public-key infrastructure for the public key $pk(a)$ and private key $sk(a)$ of agents $a \in \mathcal{V} \cup Aut$ has been set up.

- All votes have equal weight.
- The set of received ballots precisely determines the election result, and is published after the elections.
- The candidate preferred by voter is independent of the voting system, which implies that the relation between voters and their preferred candidates can be given a priori.

As the setting dictates that the set of received ballots (which determines the election result) is published after elections, counting is avoided in our framework. It is sufficient for a voting system to produce the set of received ballots.

We capture the relation between voters and their preferred candidate by the relation $\gamma: \mathcal{V} \rightarrow \mathcal{C}$. This relation specifies for each voter $v \in \mathcal{V}$ for which candidate $c \in \mathcal{C}$ she votes, viz. $\gamma(v)$. To reconstruct tuples, we use projection functions $\pi_i, i > 0$ that return the i^{th} component of a tuple. For ease of reference, $Agents = \mathcal{V} \cup Aut$.

In addition to the above, the framework is based on the following security assumptions. These assumptions follow the standard intruder model originally proposed by Dolev and Yao [DY83]:

- Cryptography works perfectly.
This assumption means that encryption and decryption are only possible with the appropriate key, that hash functions are irreversible, and that the public key infrastructure is secure (every agent knows every agent's public key, no agent knows any private key except for his own).
- Communications are under complete control of the intruder, with the exception of communications over untappable channels.
This assumption means the following: any message that is sent out, is received by the intruder. Furthermore, any received message originates from an intruder. The intruder can arbitrarily block messages, if he so chooses. As seen in Section 3.2.1, untappable channels are outside intruder control.

Note that we refer to *the* intruder instead of an intruder.

The framework expresses the interactions between agents (voters, voting authorities) as communication of terms. First we describe the syntax of terms and communication events. Then we specify the semantics of an individual agent. The semantics of the framework are then based on the semantics of the agents. We express the behaviour of the entire system as traces of events.

4.2.1 Syntax of the framework

We first define the syntax of terms, which is followed by the syntax of agents. Next, the syntactical representation of a voting system is defined as a function associating a state with each agent.

Terms

The terms communicated in a voting system are built up from variables from the set Vars , candidates from the set \mathcal{C} , random numbers from the set Nonces , and cryptographic keys from the set Keys . The set of keys of a particular agent a is given by Keys_a . These basic terms can be composed through pairing $((\varphi_1, \varphi_2))$ and encryption $(\{\varphi\}_k)$.

Definition 1 (terms). *Let Vars be a set of variables, containing at least the variable vc , let Agents be a set of agents, let \mathcal{C} be a set of candidates, let Nonces be a set of nonces, and let Keys be a set of keys, ranged over by var, a, c, n and k , respectively. The class Terms of terms, ranged over by φ , is given by the BNF*

$$\varphi ::= \text{var} \mid a \mid c \mid n \mid k \mid (\varphi_1, \varphi_2) \mid \{\varphi\}_k.$$

Syntactical equivalence of terms φ_1, φ_2 is denoted as $\varphi_1 = \varphi_2$. A term is called open if it contains variables and closed if it contains no variables. The set of variables of an open term φ is given by $\text{fv}(\varphi)$. To close an open term φ , the variables of φ are substituted by closed terms (and not bound).

Terms encrypted with k can be decrypted using the inverse key k^{-1} . For symmetric encryption, $k^{-1} = k$, whereas in asymmetric encryption, $\text{pk}(a)$ denotes the public key and $\text{sk}(a)$ the corresponding secret key of agent a . Signing is denoted as encryption with the secret key.

Example (terms) The encryption of a nonce n with the public key of agent a is denoted as $\{n\}_{\text{pk}(a)}$. A pair of this encrypted nonce with itself, unencrypted, results in the term $(\{n\}_{\text{pk}(a)}, n)$. Note that $((\{n\}_{\text{pk}(a)}, n), n) \neq (\{n\}_{\text{pk}(a)}, (n, n))$, as the terms are syntactically different.

Variables represent unspecified terms, such as a voter's choice. The voter's choice is represented by the variable vc until instantiated. Note that variables cannot be bound by a term (hence the set of variables of a term is the set of free variables of that term). However, variables can be instantiated by substitution.

Definition 2 (term substitution). *We introduce a variable mapping relation \mapsto , of type $\text{Vars} \rightarrow \text{Terms}$. The substitution of a single variable var by a term φ is denoted as $\text{var} \mapsto \varphi$. Application of a substitution σ to φ is denoted as $\sigma(\varphi)$. The domain of σ , notation $\text{dom}(\sigma)$, is the set of variables for which σ specifies a substitution. The range of σ , notation $\text{rng}(\sigma)$, is defined as $\{\sigma(\text{var}) \mid \text{var} \in \text{dom}(\sigma)\}$. The composition of two substitutions σ' after σ is denoted by $\sigma' \circ \sigma$. The empty substitution is denoted as \emptyset .*

Agents communicating terms can expect to receive a term with a certain structure, for example, a term encrypted with the agent's public key $(\{\varphi\}_{\text{pk}(a)})$. This is expressed by specifying an open term for the receiving agent (such as $\{\text{var}\}_{\text{pk}(a)}$), that serves as a template for the expected term.

Definition 3 (term matching). *A closed term φ_{cl} matches a term φ_o , if there is a substitution σ , such that $\text{dom}(\sigma) = \text{fv}(\varphi_o)$ and $\sigma(\varphi_o) = \varphi_{cl}$. We denote this as*

$$\text{match}(\varphi_{cl}, \varphi_o, \sigma) \equiv \sigma(\varphi_o) = \varphi_{cl} \wedge \text{dom}(\sigma) = \text{fv}(\varphi_o).$$

Lemma 1. (unique substitution). *Given a closed term φ_{cl} and a term φ_o , there is at most one substitution σ such that $\text{match}(\varphi_{cl}, \varphi_o, \sigma)$.*

Proof. (Proof by contradiction) Suppose that for two terms φ_{cl}, φ_o there exist σ_1, σ_2 such that:

- $\text{match}(\varphi_{cl}, \varphi_o, \sigma_1)$ and $\text{match}(\varphi_{cl}, \varphi_o, \sigma_2)$ both hold, while
- $\sigma_1 \neq \sigma_2$, $\sigma_1(\varphi_o) = \sigma_2(\varphi_o) = \varphi_{cl}$, and $\text{dom}(\sigma_1) = \text{dom}(\sigma_2) = \text{fv}(\varphi_o)$.

As $\text{dom}(\sigma_1) = \text{dom}(\sigma_2)$ and $\sigma_1 \neq \sigma_2$, there must be at least one variable $\text{var} \in \text{dom}(\sigma_1)$ such that $\sigma_1(\text{var}) \neq \sigma_2(\text{var})$. But since $\text{dom}(\sigma_1) = \text{fv}(\varphi_o)$, $\text{var} \in \text{fv}(\varphi_o)$. Thus $\sigma_1(\varphi_o) \neq \sigma_2(\varphi_o)$, which is a contradiction. \square

Example (variable matching and substitution) Consider the term of the previous example, $\{n\}_{pk(a)}$. In Table 4.1, we compare this term to several other terms to see if they match under the stated substitution. In Table 4.1, we assume that $n \neq n'$ and $pk(a) \neq k$.

φ	σ	$\text{match}(\{n\}_{pk(a)}, \varphi, \sigma)$
var	$\text{var} \mapsto \{n\}_{pk(a)}$	true
$\{\text{var}\}_{pk(a)}$	$\text{var} \mapsto n$	true
$\{n\}_{pk(a)}$	\emptyset	true
$\{n'\}_{pk(a)}$	any	false
$\{n\}_k$	any	false

Table 4.1: Example: matching and substitution of terms

A term φ may be derived from a set of terms K (notation $K \vdash \varphi$) if it is an element of K or if it can be derived by repeated application of the following rules:

(pairing)	$K \vdash \varphi_1, K \vdash \varphi_2$	$\implies K \vdash (\varphi_1, \varphi_2)$
(left)	$K \vdash (\varphi_1, \varphi_2)$	$\implies K \vdash \varphi_1$
(right)	$K \vdash (\varphi_1, \varphi_2)$	$\implies K \vdash \varphi_2$
(encryption)	$K \vdash \varphi, K \vdash k$	$\implies K \vdash \{\varphi\}_k$
(decryption)	$K \vdash \{\varphi\}_k, K \vdash k^{-1}$	$\implies K \vdash \varphi$

An agent's knowledge is a set of terms closed under derivability. Closure of a set K under derivability is defined as $\overline{K} = \{\varphi \mid K \vdash \varphi\}$.

Example (derivability and knowledge) From a knowledge set K containing term n , we can derive the following terms: n , (n, n) , $((n, n), n)$, $(n, (n, n))$, and so on. If K also contains a key k , we can additionally derive terms such as $\{n\}_k$, $(n, \{n\}_k)$, (n, k) , (k, n) , \dots

Agents

Terms are communicated between agents. Communication of a term is an event. Following the conclusions of the survey of voting systems in Section 3.2.1, we distinguish public, anonymous, and untappable communication channels. Hence, there are distinct events for each type of channel. Furthermore, we note that any election process inherently has multiple phases. As such, synchronisation between the election officials must be inherent in voting systems. For the framework to support this, we introduce a phase synchronisation event.

Definition 4 (events). *The class Ev of communication events, ranged over by ev , is given by:*

$$Ev = \{ s(a, a', \varphi), r(a, a', \varphi), as(a, a', \varphi), ar(a', \varphi), us(a, a', \varphi), ur(a, a', \varphi), \\ ph(i) \mid a, a' \in Agents, \varphi \in Terms, i \in \mathbb{N} \},$$

where s, r, as, ar, us, ur denote sending and receiving over public, anonymous and untappable channels, respectively. Finally, $ph(i)$ is the event denoting that an agent is ready to execute phase transition i .

The subclass of send events is denoted as:

$$Ev_{snd} = \{ s(a, a', \varphi), as(a, a', \varphi), us(a, a', \varphi) \mid a, a' \in Agents, \varphi \in Terms \}.$$

Similarly, the subclass of receive events is denoted as:

$$Ev_{rcv} = \{ r(a, a', \varphi), ar(a', \varphi), ur(a, a', \varphi) \mid a, a' \in Agents, \varphi \in Terms \}.$$

The subclass of phase events is denoted as $Ev_{ph} = \{ ph(i) \mid i \in \mathbb{N} \}$.

Variable substitution is extended straightforwardly to events, by replacing all substituted variables in the term of one event. This is denoted as $\sigma(ev)$ for an event ev and a substitution σ . The function fv is similarly extended, by application to the term of an event. The set of variables of an event ev is thus given by $fv(ev)$.

The behaviour of an agent is determined by the order in which events occur. This order is defined by the agent's process, which is specified in the style of process algebra's such as ACP [BW90] and μ CRL [BFG⁺01].

Definition 5 (processes). *The class $Processes$ of processes, ranged over by P , is defined as follows.*

$$P ::= \delta \mid ev.P \mid P_1 + P_2 \mid P_1 \triangleleft \varphi_1 = \varphi_2 \triangleright P_2 \mid ev.X(\varphi_1, \dots, \varphi_n).$$

Here, δ denotes a deadlock. A process can be preceded by an event ($ev.P$). Furthermore, a process can be a choice between two alternative processes, either a non-deterministic choice ($P_1 + P_2$) or a conditional choice ($P_1 \triangleleft \varphi_1 = \varphi_2 \triangleright P_2$). If φ_1 is syntactically equal to φ_2 , the process behaves as P_1 ; otherwise, the process behaves as P_2 . Finally, we have guarded recursion of processes. We assume a class of process variables, which is ranged over by X . For every process variable X , with arity n , there is a defining equation of the form $X(\mathbf{var}_1, \dots, \mathbf{var}_n) = P$, with the syntactic requirement that the free variables of P (as defined below) are precisely $\mathbf{var}_1, \dots, \mathbf{var}_n$.

Example (processes) The following are examples of processes.

$$\begin{aligned} &\delta \\ &ph(1).\delta \\ &ph(2).\delta \triangleleft \{\varphi\}_k = \{\varphi'\}_k \triangleright ph(1).\delta \\ &ph(1).X(\varphi_1, \{\varphi_2\}_k, n) \end{aligned}$$

Without loss of generality, we assume a naming convention such that all free variables in the defining equation of a process variable have globally unique names. This limits the scope of such variables to that defining equation.

Substitutions are extended to processes. The substitution σ applied to process P is denoted as $\sigma(P)$ and is defined as follows.

$$\sigma(P) = \begin{cases} \delta & \text{if } P = \delta \\ \sigma(ev).\sigma(P') & \text{if } P = ev.P' \\ \sigma(P_1) + \sigma(P_2) & \text{if } P = P_1 + P_2 \\ \sigma(P_1) \triangleleft \sigma(\varphi_1) = \sigma(\varphi_2) \triangleright \sigma(P_2) & \text{if } P = P_1 \triangleleft \varphi_1 = \varphi_2 \triangleright P_2 \\ \sigma(ev).Y(\sigma(\varphi_2), \dots, \sigma(\varphi_n)) & \text{for fresh } Y(\mathbf{var}_1, \dots, \mathbf{var}_n) = \sigma(P') \\ & \text{if } P = ev.X(\varphi_1, \dots, \varphi_n) \wedge X(\mathbf{var}_1, \dots, \mathbf{var}_n) = P' \end{cases}$$

Note that for guarded recursion (i.e. $ev.X(\varphi_1, \dots, \varphi_n)$), the substitution σ is applied to the invoked process too (i.e. to P , if $X(\mathbf{var}_1, \dots, \mathbf{var}_n) = P$). This is done by introducing a new defining equation for a fresh process variable Y . The defining equation is of the form $Y(\mathbf{var}_1, \dots, \mathbf{var}_n) = P'$, where P' is the substitution σ applied to process P , so $Y(\mathbf{var}_1, \dots, \mathbf{var}_n) = \sigma(P)$.

The function fv is also extended to processes. Note that receiving a term binds the received variables, hence receive actions reduce the number of free variables. Furthermore, remark that in guarded recursion all free variables of the defined process are bound by the invocation (due to the syntactic requirement). Thus, the only free variables of an invocation are the free variables of the argument list.

$$fv(P) = \begin{cases} \emptyset & \text{if } P = \delta \\ fv(P') & \text{if } P = ev.P' \wedge ev \in Ev_{ph} \\ fv(P') \cup fv(ev) & \text{if } P = ev.P' \wedge ev \in Ev_{snd} \\ fv(P') \setminus fv(ev) & \text{if } P = ev.P' \wedge ev \in Ev_{rcv} \\ fv(P_1) \cup fv(P_2) & \text{if } P = P_1 + P_2 \\ fv(P_1) \cup fv(\varphi_1) \cup fv(\varphi_2) \cup fv(P_2) & \text{if } P = P_1 \triangleleft \varphi_1 = \varphi_2 \triangleright P_2 \\ fv(\varphi_1) \cup \dots \cup fv(\varphi_n) & \text{if } P = ev.X(\varphi_1, \dots, \varphi_n) \\ & \wedge ev \in Ev_{ph} \\ fv(\varphi_1) \cup \dots \cup fv(\varphi_n) \cup fv(ev) & \text{if } P = ev.X(\varphi_1, \dots, \varphi_n) \\ & \wedge ev \in Ev_{snd} \\ fv(\varphi_1) \cup \dots \cup fv(\varphi_n) \setminus fv(ev) & \text{if } P = ev.X(\varphi_1, \dots, \varphi_n) \\ & \wedge ev \in Ev_{rcv} \end{cases}$$

Variables used in a defining equation which do not appear in the argument list of the equation must be bound by their occurrence. This means that such variables may only occur in events which bind variables, i.e. events $\in Ev_{rcv}$.

An agent's state is a combination of behaviour, i.e. the order in which events are executed (as determined by a process), and knowledge (a set of terms) as follows.

Definition 6 (agent state). *The state of an agent a is given by the agent's knowledge knw_a and its process P_a . Thus, agent state $st \in Agstate$ is a tuple of knowledge and a process:*

$$Agstate = \mathcal{P}(\text{Terms}) \times \text{Processes}.$$

A voting system specifies, for each agent, its state. Hence, a voting system is a mapping from agent to states.

Definition 7 (voting system). *A voting system is a system that specifies a state for each agent a . The class of voting systems, $VotSys$, is defined as $VotSys = Agents \rightarrow Agstate$. We denote the state assigned by voting system $\mathcal{VS} \in \text{VotSys}$ to agent a as $\mathcal{VS}(a) = (knw_a, P_a)$. Here, knw_a is the knowledge of agent a , and P_a describes its behaviour.*

A voting system $\mathcal{VS} \in \text{VotSys}$ may be instantiated with voter choices, as given by choice function $\gamma: \mathcal{V} \rightarrow \mathcal{C}$. This instantiation is denoted as \mathcal{VS}^γ , which, for each voter, substitutes the voter choice variable vc by the choice specified by γ in her process, as follows.

$$\mathcal{VS}^\gamma(a) = \begin{cases} \mathcal{VS}(a) & \text{if } a \notin \mathcal{V} \\ (\pi_1(\mathcal{VS}(a)), \pi_2(\sigma(\mathcal{VS}(a)))) & \text{if } a \in \mathcal{V} \wedge \sigma = \mathbf{vc} \mapsto \gamma(a) \end{cases}$$

Recall that π_i denotes a projection function that extracts the i^{th} component from a tuple.

Example (voting system) In Table 4.2 we specify a voting system \mathcal{VS}_0 with two voters va, vb and one authority T . In \mathcal{VS}_0 , voter va sends her signed vote, encrypted with T 's public key, to counter T . Voter vb does the same for her vote. Note that these votes are specified as vc , until instantiated by a choice function. The counter receives these votes in variables \mathbf{var}_1 and \mathbf{var}_2 , respectively. The initial knowledge of a voter consists of her private key and the public key of T . As there are no voter-to-voter interactions, we omit the public keys of other voters in the initial knowledge of a voter in this example. In addition to \mathcal{VS}_0 , we also show $\mathcal{VS}_0^{\gamma_1}$, where $\gamma_1(va) = ca$ and $\gamma_1(vb) = cb$.

$\mathcal{VS}_0(va)$	$= (\{pk(T), sk(va)\}, s(va, T, \{\{\mathbf{vc}\}_{sk(va)}\}_{pk(T)}) \cdot \delta)$
$\mathcal{VS}_0(vb)$	$= (\{pk(T), sk(vb)\}, s(vb, T, \{\{\mathbf{vc}\}_{sk(vb)}\}_{pk(T)}) \cdot \delta)$
$\mathcal{VS}_0(T)$	$= (\{pk(va), pk(vb), sk(T)\},$ $r(va, T, \{\{\mathbf{var}_1\}_{sk(va)}\}_{pk(T)}) \cdot r(vb, T, \{\{\mathbf{var}_2\}_{sk(vb)}\}_{pk(T)}) \cdot \delta)$
$\mathcal{VS}_0^{\gamma_1}(va)$	$= (\{pk(T), sk(va)\}, s(va, T, \{\{ca\}_{sk(va)}\}_{pk(T)}) \cdot \delta)$
$\mathcal{VS}_0^{\gamma_1}(vb)$	$= (\{pk(T), sk(vb)\}, s(vb, T, \{\{cb\}_{sk(vb)}\}_{pk(T)}) \cdot \delta)$
$\mathcal{VS}_0^{\gamma_1}(T)$	$= \mathcal{VS}_0(T)$

Table 4.2: Example: voting system \mathcal{VS}_0 and $\mathcal{VS}_0^{\gamma_1}$ (where $\gamma_1(va) = ca$ and $\gamma_1(vb) = cb$)

4.2.2 Semantics of the framework

The operational semantics of a voting system is defined by a number of derivation rules of the form

$$\frac{p_1 \dots p_n}{S \xrightarrow{e} S'}.$$

This expresses that if the system is in state S and if the premises p_1 to p_n are satisfied, the system may perform event e and continue in state S' (see e.g. [Plo81, Plo04, BV96]). The operational semantics is defined in two layers. First, the semantics of individual agents is defined. Next we define the semantics of a voting system based on the semantics of individual agents. The operational semantics of a voting system can be seen as the parallel composition of all agents.

Agent semantics

The semantics of agents describes the effect of the events on the agent state. Recall that agent state is defined as a tuple containing a knowledge set and a process: $Agstate = \mathcal{P}(Terms) \times Processes$.

In our assumed intruder model, each tappable communication by an agent is a communication with the intruder. Hence, the semantic rules below take the intruder's knowledge into account. The states considered below thus consist of a tuple of intruder knowledge K_I and agent state. In the rules below, events may involve $a, x \in Agents$, which we omit from the premise of the rules.

There are some restrictions on the terms that may occur in an event. A term φ occurring in a send event must be closed ($fv(\varphi) = \emptyset$) at the moment of sending. A term φ occurring in a receive event can specify the structure of the term φ' to be received. There is a limitation: a receiving agent a can only specify the structure up to the extent of his knowledge knw_a . The readable predicate specifies if an agent a can receive φ_{cl} using a term φ_o as a receive pattern, for a substitution σ such that $match(\varphi_{cl}, \varphi_o, \sigma)$ holds, as follows.

Definition 8 (readability of terms). *The readability of a term φ depends on those terms φ' necessary to deconstruct φ . We denote this as $\varphi' \sqsubseteq \varphi$. The deconstruction operator \sqsubseteq is defined inductively as follows (cf. [Cre06]).*

$$\begin{array}{ll} \varphi \sqsubseteq \varphi & \\ \varphi_1 \sqsubseteq (\varphi_1, \varphi_2) & \varphi_2 \sqsubseteq (\varphi_1, \varphi_2) \\ \varphi \sqsubseteq \{\varphi\}_k & k^{-1} \sqsubseteq \{\varphi\}_k \end{array}$$

Note that to deconstruct an encryption, the decryption key is required.

Using this operator, agent a 's ability to read a term φ_{cl} using a term φ_o acting as a pattern, given a suitable substitution σ , is defined as follows.

$$Rd(knw_a, \varphi_{cl}, \varphi_o, \sigma) \equiv match(\varphi_{cl}, \varphi_o, \sigma) \wedge \forall \varphi' \sqsubseteq \varphi_o : knw_a \cup \{\varphi_{cl}\} \vdash \sigma(\varphi').$$

Thus, a term φ_{cl} is readable for agent a with pattern φ_o and substitution σ , if for every subterm of φ_o , the instantiation of that subterm is derivable from the knowledge of agent a enriched with φ_{cl} .

Example (readability of terms) Below, we illustrate the readability function for several cases.

φ	p	knw	σ	$\text{Rd}(knw, \varphi, p, \sigma)$
n	var	\emptyset	$\text{var} \mapsto n$	<i>true</i>
n	n	$\{n\}$	\emptyset	<i>true</i>
$\{n\}_k$	$\{\text{var}\}_k$	\emptyset	$\text{var} \mapsto n$	<i>false</i>
$\{n\}_k$	$\{\text{var}\}_k$	$\{k^{-1}\}$	$\text{var} \mapsto n$	<i>true</i>
(φ_1, φ_2)	var	\emptyset	$\text{var} \mapsto (\varphi_1, \varphi_2)$	<i>true</i>
(φ_1, φ_2)	(var, var)	\emptyset	$\text{var} \mapsto \varphi_1$	$\varphi_1 = \varphi_2$

This definition allows us to specify the operational semantics of agent behaviour as follows.

public send. An agent may send a closed term φ if and only if the agent knows the term (note that variables do not capture knowledge).

$$\frac{knw_a \vdash \varphi \quad \text{fv}(\varphi) = \emptyset}{(K_I, knw_a, s(a, x, \varphi).P) \xrightarrow{s(a, x, \varphi)} (K_I \cup \{\varphi\}, knw_a, P)}$$

public receive. A receive event specifies an open, readable term φ . By receiving a matching term φ' , the unassigned variables in φ are assigned a value by the unique substitution σ such that $\text{Rd}(knw_a, \varphi', \varphi, \sigma)$.

$$\frac{K_I \vdash \varphi' \quad \text{fv}(\varphi') = \emptyset \quad \text{Rd}(knw_a, \varphi', \varphi, \sigma)}{(K_I, knw_a, r(x, a, \varphi).P) \xrightarrow{r(x, a, \varphi')} (K_I, knw_a \cup \{\varphi'\}, \sigma(P))}$$

anonymous send. The anonymous send semantics follows directly from the above public send semantics by replacing the send event $s(x, y, \varphi)$ with event $as(x, y, \varphi)$.

$$\frac{knw_a \vdash \varphi \quad \text{fv}(\varphi) = \emptyset}{(K_I, knw_a, as(a, x, \varphi).P) \xrightarrow{as(a, x, \varphi)} (K_I \cup \{\varphi\}, knw_a, P)}$$

anonymous receive. An anonymous receive is largely equal to the public receive, except that the event omits information about the sender.

$$\frac{K_I \vdash \varphi' \quad \text{fv}(\varphi') = \emptyset \quad \text{Rd}(knw_a, \varphi', \varphi, \sigma)}{(K_I, knw_a, ar(a, \varphi).P) \xrightarrow{ar(a, \varphi')} (K_I, knw_a \cup \{\varphi'\}, \sigma(P))}$$

untappable send. An untappable send is a send event which happens outside intruder control or even intruder awareness. It thus limits the power of the intruder. Note that the intruder does not learn the communicated term for an untappable send event.

$$\frac{knw_a \vdash \varphi \quad \text{fv}(\varphi) = \emptyset}{(K_I, knw_a, us(a, x, \varphi).P) \xrightarrow{us(a, x, \varphi)} (K_I, knw_a, P)}$$

untappable receive. An untappable receive is the receiving dual of the untappable send. Thus, it occurs outside intruder control or awareness, and represents a limit on the power of the intruder. Note that, at this level of the semantics, the origin of the term φ' is not specified. The origin of this term is specified at the system level, where untappable receive is synchronised with untappable send.

$$\frac{\text{fv}(\varphi') = \emptyset \quad \text{Rd}(\text{knw}_a, \varphi', \varphi, \sigma)}{(K_I, \text{knw}_a, \text{ur}(x, a, \varphi).P) \xrightarrow{\text{ur}(x, a, \varphi')} (K_I, \text{knw}_a \cup \{\varphi'\}, \sigma(P))}$$

phase synchronisation. The events of the set Ev_{ph} are intended to synchronise agents in the system. At the agent level, these events have little impact, as evidenced by the following operational semantic rule.

$$\frac{ev \in Ev_{ph}}{(K_I, \text{knw}_a, ev.P) \xrightarrow{ev} (K_I, \text{knw}_a, P)}$$

non-deterministic choice. An agent that can execute a non-deterministic choice may choose any of the alternatives, as follows.

$$\frac{(K_I, \text{knw}_a, P_1) \xrightarrow{ev} (K'_I, \text{knw}'_a, P'_1)}{(K_I, \text{knw}_a, P_1 + P_2) \xrightarrow{ev} (K'_I, \text{knw}'_a, P'_1)}$$

$$\frac{(K_I, \text{knw}_a, P_2) \xrightarrow{ev} (K'_I, \text{knw}'_a, P'_2)}{(K_I, \text{knw}_a, P_1 + P_2) \xrightarrow{ev} (K'_I, \text{knw}'_a, P'_2)}$$

conditional choice. A conditional choice is a choice between two processes, based upon syntactical comparison of two terms. While these two terms may be open terms in the agent's specification, upon execution, we require that these terms are closed.

$$\frac{(K_I, \text{knw}_a, P_1) \xrightarrow{ev} (K'_I, \text{knw}'_a, P'_1) \quad \text{fv}(\varphi_1) = \emptyset}{(K_I, \text{knw}_a, P_1 \triangleleft \varphi_1 = \varphi_1 \triangleright P_2) \xrightarrow{ev} (K'_I, \text{knw}'_a, P'_1)}$$

$$\frac{(K_I, \text{knw}_a, P_2) \xrightarrow{ev} (K'_I, \text{knw}'_a, P'_2) \quad \varphi_1 \neq \varphi_2 \quad \text{fv}(\varphi_1) = \text{fv}(\varphi_2) = \emptyset}{(K_I, \text{knw}_a, P_1 \triangleleft \varphi_1 = \varphi_2 \triangleright P_2) \xrightarrow{ev} (K'_I, \text{knw}'_a, P'_2)}$$

guarded recursion. Agent a may execute an invocation of process variable X with argument list $\varphi_1, \dots, \varphi_n$ if the corresponding process in the defining equation can execute, under the specified arguments.

$$\frac{X(\text{var}_1, \dots, \text{var}_n) = P \quad \sigma = \text{var}_1 \mapsto \varphi_1 \circ \dots \circ \text{var}_n \mapsto \varphi_n \quad (K_I, \text{knw}_a, \sigma(P)) \xrightarrow{ev} (K'_I, \text{knw}'_a, P')}{(K_I, \text{knw}_a, X(\varphi_1, \dots, \varphi_n)) \xrightarrow{ev} (K'_I, \text{knw}'_a, P')}$$

System semantics

The operational semantics of voting systems describe how the state of a voting system changes due to the interactions of its agents. The state of a voting system is given by the intruder knowledge and the state of each agent.

Definition 9 (state of a voting system). *The state of a voting system is a tuple of intruder knowledge and a mapping of agents to agent states (recall that $\text{VotSys} = \text{Agents} \rightarrow \text{Agstate}$), as follows.*

$$\text{State} = \mathcal{P}(\text{Terms}) \times \text{VotSys}.$$

The knowledge and current process for each agent are given by VotSys . We denote the attribution of state (knw_a, P_a) to agent a as $a @ (\text{knw}_a, P_a)$. The current state of agent a in system state (K_I, S) is denoted as $a @ (\text{knw}_a, P_a) \in S$. The initial state of voting system \mathcal{VS} with respect to choice function γ is $(K_I^0, \mathcal{VS}^\gamma)$, for initial intruder knowledge K_I^0 .

Typically, the initial intruder knowledge contains public keys of all agents, compromised keys etc.

The operational semantics of voting systems in the context of a Dolev-Yao intruder (limited to public and anonymous channels) is given below. The semantic rules give rise to a labelled transition system, with labels denoting the events. Untappable communication is modelled as synchronous communication, hence the *us* and *ur* events are replaced by *uc* events (denoting untappable communication) in the set of labels *Labels* of the transition system:

$$\begin{aligned} \text{Labels} = & \{uc(a, a', \varphi) \mid a, a' \in \text{Agents} \wedge \varphi \in \text{Terms}\} \cup \\ & Ev \setminus \{us(a, a', \varphi), ur(a, a', \varphi) \mid a, a' \in \text{Agents} \wedge \varphi \in \text{Terms}\}. \end{aligned}$$

Both untappable communications and phase synchronisation are synchronous events by more than one agent. The other events are executed without synchronising with other agents. We distinguish the non-synchronous events as Ev_{nosync} , which is defined as follows.

$$Ev_{\text{nosync}} = \{s(a, a', \varphi), r(a, a', \varphi), as(a, a', \varphi), ar(a', \varphi) \mid a, a' \in \text{Agents}, \varphi \in \text{Terms}\}.$$

The below system semantics uses the above agent semantics to define the dynamic behaviour of the system. The rules below may involve agents $a, b \in \text{Agents}$, which we omit from the premises of the rules. Note that the premise of the rules involves agent state transitions (a three-tuple of intruder knowledge, agent knowledge and agent process), and may specify restrictions on the system state (a mapping of agents to agent states).

non-synchronous events. The operational semantics for public and for anonymous send as well as read events is given by the following rule.

$$\frac{(K_I, \text{knw}_a, P) \xrightarrow{ev} (K'_I, \text{knw}'_a, P') \quad ev \in Ev_{\text{nosync}} \quad a @ (\text{knw}_a, P) \in S}{(K_I, S) \xrightarrow{ev} (K'_I, \{a @ (\text{knw}'_a, P')\} \cup S \setminus \{a @ (\text{knw}_a, P)\})}$$

untappable communications. As no agent, nor the intruder, except for the sending agent and the receiving agent, are aware of untappable communications, we model them as synchronous communication. This captures both untappable send and receive in one transition. Hence, a new label for this transition is needed. We use $uc(a, b, \varphi)$, which must match both the send $us(a, b, \varphi)$ and the receive $ur(a, b, \varphi)$ events. Note that the intruder's knowledge does not change due to untappable communications, nor does the sending agent's knowledge.

$$\frac{\begin{array}{c} (K_I, knw_a, P_a) \xrightarrow{us(a, b, \varphi)} (K_I, knw_a, P'_a) \\ (K_I, knw_b, P_b) \xrightarrow{ur(a, b, \varphi)} (K_I, knw'_b, P'_b) \\ s_0 = \{a @ (knw_a, P_a), b @ (knw_b, P_b)\} \quad s_0 \subseteq S \end{array}}{(K_I, S) \xrightarrow{uc(a, b, \varphi)} (K_I, \{a @ (knw_a, P'_a), b @ (knw'_b, P'_b)\} \cup S \setminus s_0)}$$

phase synchronisation. Phase events denote synchronisation points. A $ph(i)$ event may only occur if all authorities have agreed that the election will evolve into a new phase. As a consequence, all agents who are ready and willing to do so will move to the new phase as well.¹

In the semantics rule below, the agents that are ready and willing to execute the phase transition are captured (together with their states) in the set $Phase$. Note that $Phase$ is a subset of all agents ready to execute a phase transition, as readiness does not imply willingness. The set $Phase'$ reflects the new states for these agents. Finally, we explicitly require each authority $a \in Aut$ to be ready and willing to execute the phase transition.

$$\frac{\begin{array}{c} i \in \mathbb{N} \\ Phase \subseteq \{a @ (knw_a, P_a) \in S \mid \exists P'_a: (K_I, knw_a, P_a) \xrightarrow{ph(i)} (K_I, knw_a, P'_a)\} \\ Aut \subseteq \{a \in Agents \mid \exists knw_a, P_a: a @ (knw_a, P_a) \in Phase\} \\ Phase' = \{a @ (knw_a, P'_a) \mid \exists P_a: a @ (knw_a, P_a) \in Phase\} \\ (K_I, knw_a, P_a) \xrightarrow{ph(i)} (K_I, knw_a, P'_a) \end{array}}{(K_I, S) \xrightarrow{ph(i)} (K_I, Phase' \cup S \setminus Phase)}$$

The above semantics rules give rise to labelled transition systems. Each possible execution of the system is represented by a path in this labelled transition system. A path is represented by a list of its labels, which is called a trace. The set of traces of a given voting system is defined as follows.

Definition 10 (traces). *The class of traces $Traces$ consists of finite lists of labels. The traces of a voting system \mathcal{VS}^γ (voting system \mathcal{VS} instantiated with choice function γ) are given by*

$$\begin{aligned} Tr(\mathcal{VS}^\gamma) = \{ & \alpha \in Labels^* \mid \alpha = \alpha_0 \dots \alpha_{n-1} \wedge \\ & \exists s_0, \dots, s_n \in State: s_0 = (K_I^0, \mathcal{VS}^\gamma) \wedge \\ & \forall 0 \leq i < n: s_i \xrightarrow{\alpha_i} s_{i+1} \} \end{aligned}$$

¹We conjecture that our semantics of phase synchronisation is similar to the strong phase semantics as proposed in [DRS08].

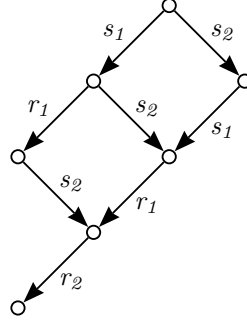


Figure 4.1: A labelled transition system

The set of traces of a voting system \mathcal{VS} is now given by

$$Tr(\mathcal{VS}) = \bigcup_{\gamma \in \mathcal{V} \rightarrow \mathcal{C}} Tr(\mathcal{VS}^\gamma).$$

We denote the intruder knowledge in the last state of a trace t as K_I^t . The empty trace is denoted by ϵ .

Example (traces) Consider the voting system \mathcal{VS}_0 from Table 4.2, and a choice function γ_1 such that $\gamma_1(va) = \gamma_1(vb) = c$. Then we have the following.

$$\begin{aligned} \epsilon &\in Tr(\mathcal{VS}_0^{\gamma_1}) \\ s(va, T, \{\{c\}_{sk(va)}\}_{pk(T)}) &\in Tr(\mathcal{VS}_0^{\gamma_1}) \\ s(va, T, \{\{c\}_{sk(va)}\}_{pk(T)}) \ s(vb, T, \{\{c\}_{sk(vb)}\}_{pk(T)}) &\in Tr(\mathcal{VS}_0^{\gamma_1}) \\ s(va, T, \{\{c\}_{sk(va)}\}_{pk(T)}) \ r(va, T, \{\{c\}_{sk(va)}\}_{pk(T)}) &\in Tr(\mathcal{VS}_0^{\gamma_1}) \\ \dots & \end{aligned}$$

If we denote the send event of voter va as s_1 , the send event of voter vb as s_2 and the corresponding read events of the tallier as r_1, r_2 , respectively, the labelled transition system for this voting system is as in Figure 4.1. Its full set of traces is succinctly described as

$$Tr(\mathcal{VS}_0^{\gamma_1}) = \{\epsilon, s_1, s_2, s_1 s_2, s_2 s_1, s_1 r_1, s_1 s_2 r_1, s_2 s_1 r_1, s_1 r_1 s_2, s_1 s_2 r_1 r_2, s_1 r_1 s_2 r_2, s_2 s_1 r_1 r_2\}.$$

Traces model the dynamic behaviour of the system. The next section determines the privacy of a given voter in a given trace. This trace-based definition of voter privacy is then extended to establish the privacy of a voter in a voting system.

4.3 Privacy in voting systems

The framework developed in the previous section enables us to express if an intruder can distinguish two executions of the system, as previously expressed by Mauw, Verschuren and De Vink [MVV04] and later by Garcia et al. [GHPR05] for passive intruders. Traces t, t' are to be considered equivalent if the intruder cannot distinguish

them. To formalise this equivalence, the distinguishing ability of the intruder is formalised as the intruder's ability to distinguish two messages. We introduce the notion of *reinterpretation* to capture this.

Definition 11 (reinterpretation [GHPR05]). *Let ρ be a permutation on the set of terms $Terms$ and let K_I be a knowledge set. Map ρ is called a semi-reinterpretation under K_I if it satisfies the following.*

$$\begin{aligned} \rho(\varphi) &= \varphi & \text{for } \varphi \in \mathcal{C} \cup \{a, sk(a), pk(a) \mid a \in Agents\} \\ \rho((\varphi_1, \varphi_2)) &= (\rho(\varphi_1), \rho(\varphi_2)) \\ \rho(\{\varphi\}_k) &= \{\rho(\varphi)\}_{\rho(k)} & \text{if } K_I \vdash \varphi, k \vee K_I \vdash \{\varphi\}_k, k^{-1} \end{aligned}$$

Map ρ is a reinterpretation under K_I iff it is a semi-reinterpretation and its inverse ρ^{-1} is a semi-reinterpretation under $\rho(K_I)$. The notion of reinterpretation is extended straightforwardly to events and to traces by applying ρ to the message fields of events (in traces).

The notion of reinterpretation models the intruder's ability to distinguish terms. The intruder can distinguish any candidate and agent name from any other term. As public keys and private keys identify the agent, these too can be distinguished from any other term. As the intruder does not know which nonces or (non-public / private) keys belong to which agents, he cannot distinguish these from other terms. Furthermore, the intruder can distinguish the structure of paired terms. Encrypted terms can only be distinguished if the intruder can decrypt the term or if he can construct the term himself. Note that as the intruder cannot distinguish one key from another, ρ must be applied to the encryption key of any encrypted message that he can distinguish.

Example (reinterpretation) In Table 4.3, we provide two permutations on terms ρ and ρ' that are reinterpretations for $k, k', n, n', n'' \in K_I$.

φ	$\rho(\varphi)$	$\rho'(\varphi)$
n	n'	n''
k	k'	k
c	c	c
$\{(c, n)\}_k$	$\{(c, n')\}_{k'}$	$\{c, n''\}_k$
$sk(a)$	$sk(a)$	$sk(a)$

Table 4.3: Example: reinterpretation of terms

Some events in a trace are hidden from the intruder, hence the intruder has a restricted view of a trace. In particular, the intruder cannot see any *uc* transitions (communications over untappable channels), nor the sender of anonymous communications. The visible part of a trace is captured by the function $obstr: Traces \rightarrow Traces$ as follows:

$$\begin{aligned} obstr(\epsilon) &= \epsilon \\ obstr(\ell \cdot t) &= \begin{cases} obstr(t) & \text{if } \ell = uc(a, a', \varphi) \\ as(x, \varphi) \cdot obstr(t) & \text{if } \ell = as(a, x, \varphi) \\ \ell \cdot obstr(t) & \text{otherwise} \end{cases} \end{aligned}$$

Definition 12 (trace indistinguishability). *Traces t, t' are indistinguishable for the intruder, notation $t \sim t'$ iff there exists a reinterpretation ρ such that the visible part of t is the visible part of $\rho(t')$ and the final intruder knowledge in t and t' is equal modulo ρ . Formally put:*

$$t \sim t' \equiv \exists \rho: \text{obstr}(t') = \rho(\text{obstr}(t)) \wedge \overline{K_I^t} = \rho(\overline{K_I^{t'}}).$$

The above definition of the intruder's ability to distinguish traces extends to his ability to distinguish sets of traces as follows.

Definition 13 (choice indistinguishability). *Given voting system \mathcal{VS} , choice functions γ_1, γ_2 are indistinguishable to the intruder, notation $\gamma_1 \simeq_{\text{vs}} \gamma_2$ iff*

$$\begin{aligned} \forall t \in \text{Tr}(\mathcal{VS}^{\gamma_1}): \exists t' \in \text{Tr}(\mathcal{VS}^{\gamma_2}): t \sim t' \quad \wedge \\ \forall t \in \text{Tr}(\mathcal{VS}^{\gamma_2}): \exists t' \in \text{Tr}(\mathcal{VS}^{\gamma_1}): t \sim t' \end{aligned}$$

The set of choice functions indistinguishable for the intruder in a given system is now succinctly defined as follows.

Definition 14 (choice group). *The choice group for a voting system \mathcal{VS} and a choice function γ is given by*

$$\text{cg}(\mathcal{VS}, \gamma) = \{\gamma' \mid \gamma \simeq_{\text{vs}} \gamma'\}.$$

The choice group for a particular voter v , i.e. the set of candidates indistinguishable from v 's chosen candidate, is given by

$$\text{cg}_v(\mathcal{VS}, \gamma) = \{\gamma'(v) \mid \gamma' \in \text{cg}(\mathcal{VS}, \gamma)\}.$$

In the last definition, the privacy of a voting system is defined with respect to an intruder who can control all communication channels except the untappable channels. The next chapter poses the question of how much of this remaining privacy is controlled by the voter.

4.4 Conspiring voters

The above framework captures the behaviour of a passive voter, who does not actively cooperate with the intruder to prove how she has voted. However, as remarked in the introduction, we focus on voters trying to renounce their vote-privacy. A conspiring voter can try to share her knowledge with the intruder. The classic receipt-freeness case assumes the voter shares her final knowledge. As noted in [JV06], the timing of knowledge sharing is important. In order to prove that she really has the receipt, the voter needs to share her private knowledge before it becomes public in the course of the execution of the voting system. Furthermore, during the course of an election, a voter may learn or commit to knowledge that the intruder is unaware of, due to untappable communications or a voting booth. A voter may seek to circumvent such privacy provisions by sharing any acquired knowledge with the intruder, and by using intruder-supplied information for any outgoing communication. As stated in the conclusions of Section 3.2.1, voting booths can be equated to a bi-directional untappable channel.

The timing of sharing information between the conspiring voter and the intruder is important. We distinguish the cases where the voter shares her full knowledge (post-election or pre-election) from the cases where the voter conspires to nullify the effect of untappable channels (sharing information, using intruder-supplied information, or both). In absence of untappable channels, all communications are visible to the intruder. In this case, the sooner a voter shares her knowledge with the intruder, the more traces the intruder can distinguish. Classical receipt-freeness, *classic-rf*, tries to break vote-privacy by sharing knowledge after elections. However, sharing knowledge beforehand, *start-rf*, gives the intruder more knowledge during the elections. This situation is depicted below in Figure 4.2(i).

In the presence of untappable channels, the intruder is not aware of every increase of the voter's knowledge. The voter can mitigate this by conspiring mid-election. Her willingness to do so is captured in Figure 4.2(ii). The conspiring voter may choose to share information the intruder would not learn otherwise (*rf-share*) or use intruder-supplied terms in communications hidden from the intruder (*rf-witness*) to later prove how she voted. The combination of these two notions is at the top of the ordering (*rf-relay*).

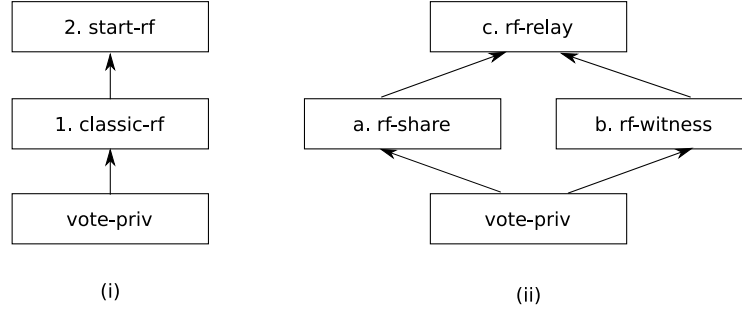


Figure 4.2: Knowledge sharing, (i) pre- and post-election and (ii) mid-election

A voter may use privacy-reducing techniques from both hierarchies to reduce her privacy. We denote this, e.g., as a *type 1a* voter, or a *type 2c* voter. In the next section, we present precise definitions of these notions.

4.5 Modelling conspiratory behaviour

A conspiring voter behaves differently from a regular voter, as she will communicate with the intruder in certain circumstances. We incorporate the different conspiracy classes into the framework as follows. We start by selecting a regular voter and the conspiracy class that we assume her to satisfy. Then the regular specification of this voter is transformed into a conspiring specification by following the transformation rules of this conspiracy class as defined below. For instance, the transformation for class 1 (*classic-rf*) consists of adding an extra event at the end of the voter's specification which represents the sharing of her knowledge with the intruder.

In order to formalise this approach, we extend the set of events Ev with events: $\{is(v, \varphi), ir(v, \varphi)\}$, where $is(v, \varphi)$ denotes agent v sending term φ to the intruder,

and $ir(v, \varphi)$ denotes agent v receiving term φ from the intruder. The agent level semantics of these events is given below:

$$\begin{array}{l}
 \text{send to intruder:} \quad \frac{knw_v \vdash \varphi \quad \text{fv}(\varphi) = \emptyset}{(K_I, knw_v, is(v, \varphi).P) \xrightarrow{is(v, \varphi)} (K_I \cup \{\varphi\}, knw_v, P)} \\
 \\
 \text{receive from intruder:} \quad \frac{K_I \vdash \varphi' \quad \text{fv}(\varphi') = \emptyset \quad \text{Rd}(knw_v, \varphi', \varphi, \sigma)}{(K_I, knw_v, ir(v, \varphi).P) \xrightarrow{ir(v, \varphi')} (K_I, knw_v \cup \{\varphi'\}, \sigma(P))}
 \end{array}$$

The specific conspiracy classes are captured by a syntactical transformation of the process of the conspiring voter v as follows:

- **1. classic-rf:** at the end of her process, v sends her knowledge set to the intruder (knw_v is represented as a pairing of each element of knw_v).
- **2. start-rf:** as the first event executed by v , she sends her knowledge set to the intruder.
- **a. rf-share:** each $ur(a, v, \varphi)$ is followed by an $is(\varphi)$.
- **b. rf-witness:** The intruder supplies the voter-controllable parts of the term used in each $us(v, a, \varphi)$. To do so, the intruder must know what terms are voter-controllable. To this end, we introduce two functions:
 - a function $\text{vars}(v, \varphi)$ that returns the variables of φ that agent v can control, and
 - a function $\text{freshvars}(v, \varphi)$, that replaces every voter-controllable variable in φ by a fresh variable.

The voter sends $\text{vars}(v, \varphi)$ to the intruder, who replies with a similar term, changing the values to his liking. The voter then uses the newly supplied values in the untappable send event.

- **c. rf-full:** this combines rf-share and rf-witness.

Note that where classic-rf and start-rf transform the ending and beginning, respectively, of voter v 's process, the other three conspiracy classes transform events inside the voter process. To model conspiracy, we first introduce an event transformation function θ_i , which relies on auxiliary functions vars and freshvars . Then, the event transformation function is then extended to processes and to voting systems.

The function $\text{vars}: \mathcal{V} \times \text{Terms} \rightarrow \text{Terms}$ captures those variables of a term φ that are under a voter v 's control. This function is defined as follows.

$$\text{vars}(v, \varphi) = \begin{cases} \{\varphi\} & \text{if } \varphi \in \text{Vars} \\ \text{vars}(v, \varphi') & \text{if } \varphi = \{\varphi'\}_k, \text{ for } k \in \text{Keys}_v \\ \text{vars}(v, \varphi_1) \cup \text{vars}(v, \varphi_2) & \text{if } \varphi = (\varphi_1, \varphi_2) \\ \emptyset & \text{otherwise} \end{cases}$$

The function $\text{freshvars}: \mathcal{V} \times \text{Terms} \rightarrow \text{Terms}$ replaces every voter-controllable variable in φ by a fresh variable. To this end, we assume the existence of a substitution σ_{fresh} , where $\text{dom}(\sigma_{\text{fresh}}) = \text{vars}(v, \varphi)$, that substitutes fresh variables for every voter-controlled variable. Note that some, but not all occurrences of a variable may be voter-controllable (e.g. var in the term $(\text{var}, \{\text{var}\}_{sk(R)})$). Hence, σ_{fresh} may only freshen those uses of variables that are under voter control, as follows.

$$\text{freshvars}(v, \varphi) = \begin{cases} \sigma_{\text{fresh}}(\text{var}) & \text{if } \varphi \in \text{Vars} \\ \{\text{freshvars}(v, \varphi')\}_k & \text{if } \varphi = \{\varphi'\}_k, \text{ for } k \in \text{Keys}_v \\ (\text{freshvars}(v, \varphi_1), \text{freshvars}(v, \varphi_2)) & \text{if } \varphi = (\varphi_1, \varphi_2) \\ \varphi & \text{otherwise} \end{cases}$$

Definition 15 (event transformation). *We define the following event transformation functions $\theta_i: \mathcal{V} \times \text{Ev} \rightarrow \text{Processes}$, for i a conspiracy class $\in \{a, b, c\}$.*

$$\begin{aligned} - \theta_a(v, ev) &= \begin{cases} \text{ur}(ag, v, \varphi) \cdot \text{is}(v, \varphi) & \text{if } ev = \text{ur}(ag, v, \varphi) \\ ev & \text{otherwise} \end{cases} \\ - \theta_b(v, ev) &= \begin{cases} \text{is}(v, \text{vars}(v, \varphi)) \cdot \text{ir}(v, \text{vars}(v, \varphi')) \cdot \text{us}(v, ag, \varphi') & \text{if } ev = \text{us}(v, ag, \varphi), \text{ for } \varphi' = \text{freshvars}(v, \varphi) \\ ev & \text{otherwise} \end{cases} \\ - \theta_c(v, ev) &= \theta_b(v, \theta_a(v, ev)). \end{aligned}$$

We model sending of a set (for example, $\text{is}(v, \text{vars}(v, \varphi))$) as sending a pairing of all elements of the set (in this example, $\text{vars}(v, \varphi)$).

The event transformation θ_a ensures that every untappable receive event is followed by an intruder send event. The event transformation θ_b ensures that the intruder supplies all voter-controllable input for every untappable send event. The event transformation θ_c combines θ_a and θ_b .

Example (event transformation) In Table 4.4, we show some examples of event rewrites. We leave out θ_c , as it is the composition of θ_a and θ_b .

ev	$\theta_a(ev)$	$\theta_b(ev)$
$ph(1)$	$ph(1)$	$ph(1)$
$\text{ur}(a, b, \text{var})$	$\text{ur}(a, b, \text{var}) \cdot \text{is}(v, \text{var})$	$\text{ur}(a, b, \text{var})$
$\text{us}(a, b, (\text{vc}, \{\text{vc}\}_{sk(R)}))$	$\text{us}(a, b, (\text{vc}, \{\text{vc}\}_{sk(R)}))$	$\text{is}(v, \text{vc}) \cdot \text{ir}(v, \text{irvar}) \cdot \text{us}(a, b, (\text{irvar}, \{\text{vc}\}_{sk(R)}))$

Table 4.4: Example: event transformation

We model a conspiring voter, according to the conspiracy classification above, by means of a process transformation function. This function transforms the process by introducing conspiring behaviour.

Definition 16 (process transformation). *The function $\Theta_i: \mathcal{V} \times \text{Processes} \rightarrow \text{Processes}$ transforms a process for a specific voter v into a conspiring process of the class $i \in \{1, 2, a, b, c\}$. For $i = 2$, conspiracy is a matter of sharing initial knowledge, which is modelled as*

$$\Theta_2(v, P) = is(knw_v).P$$

In the other cases, conspiracy has an effect on the events of the processes. For readability, we do not distinguish cases for $i = 1$, but simply state $\theta_1(v, ev) = ev$. For $i \in \{1, a, b, c\}$, process transformation is defined as follows.

$$\Theta_i(v, P) = \begin{cases} \delta & \text{if } i \neq 1 \wedge P = \delta \\ is(v, knw_v).\delta & \text{if } i = 1 \wedge P = \delta \\ \theta_i(v, ev).\Theta_i(v, P) & \text{if } P = ev.P \\ \Theta_i(v, P_1) + \Theta_i(v, P_2) & \text{if } P = P_1 + P_2 \\ \Theta_i(v, P_1) \triangleleft \varphi_1 = \varphi_2 \triangleright \Theta_i(v, P_2) & \text{if } P = P_1 \triangleleft \varphi_1 = \varphi_2 \triangleright P_2, \\ & \text{for } \varphi_1, \varphi_2 \in \text{Terms} \\ \theta_i(v, ev).Y(\varphi_1, \dots, \varphi_n), & \text{for fresh } Y(\text{var}_1, \dots, \text{var}_n) = \Theta_i(v, P') \\ & \text{if } P = X(\varphi_1, \dots, \varphi_n) \wedge X(\text{var}_1, \dots, \text{var}_n) = P' \end{cases}$$

Example (process transformation) In the example process transformations below, s_1 is the send action of voting system \mathcal{VS}_0 , in Table 4.2. The first two entries transform voter process 1 from \mathcal{VS}_0 . The final example illustrates how a more complex process is transformed.

P	i	$\Theta_i(v, P)$
$s_1.\delta$	1	$s_1.is(v, (pk(T), sk(va))).\delta$
$s_1.\delta$	2	$is(v, (pk(T), sk(va))).s_1.\delta$
$s_1.\delta + s_2.\delta$	$\in \{a, b, c\}$	$\theta_i(v, s_1).\delta + \theta_i(v, s_2).\delta$

Process transformation is extended to voting systems as follows.

$$\Theta_i(v, \mathcal{VS})(a) = \begin{cases} \mathcal{VS}(a) & \text{if } a \neq v \\ (\pi_1(\mathcal{VS}(v)), \Theta_i(v, \pi_2(\mathcal{VS}(v)))) & \text{if } a = v \end{cases}$$

The above transformations are extended to $i \in \{1a, 2a, 1b, 2b, 1c, 2c\}$ for combinations of conspiring behaviour, e.g. $\Theta_{1a}(v, \mathcal{VS}) = \Theta_1(v, \Theta_a(v, \mathcal{VS}))$. Using the above system transformation, we can define the choice group for conspiring voters in a voting system within our framework (see Section 4.3).

Definition 17 (conspiracy induced choice group). *The choice group of conspiring voter v in voting system \mathcal{VS} given choice function γ , with respect to different conspiracy classes $i \in \{1, 2, a, b, c, 1a, 2a, 1b, 2b, 1c, 2c\}$, is given by*

$$cg_v^i(\mathcal{VS}, \gamma) = cg_v(\Theta_i(v, \mathcal{VS}), \gamma).$$

Given this definition of privacy, conspiracy-resistance is a measure of the voter's choice groups as follows.

Definition 18 (conspiracy-resistance). *We call voting system \mathcal{VS} conspiracy-resistant for conspiring behaviour $i \in \{1, 2, a, b, c\}$ iff*

$$\forall v \in \mathcal{V}, \gamma \in \mathcal{V} \rightarrow \mathcal{C}: cg_v^i(\mathcal{VS}, \gamma) = cg_v(\mathcal{VS}, \gamma).$$

Remark that for when $|\mathcal{V}| = 1$ or $|\mathcal{C}| = 1$, we have $\forall \gamma: cg(\mathcal{VS}, \gamma) = \{\gamma\}$, which implies that $\forall v \in \mathcal{V}: cg_v(\mathcal{VS}, \gamma) = cg_v^i(\mathcal{VS}, \gamma) = \{\gamma(v)\}$. Thus, in such settings, there is not vote-privacy to lose, and conspiracy-resistance is satisfied trivially.

The notion of receipt-freeness as introduced by Benaloh and Tuinstra [BT94] coincides with $\forall v, \gamma: |cg_v^1(\mathcal{VS}, \gamma)| > 1$. The approach of Delaune, Kremer and Ryan [DKR06, DKR08] coincides with choice groups of size greater than one. Our framework captures any modifier of privacy, including modifiers that are not privacy-nullifying. The amount of conspiracy-resistance of a system is measured as the difference in privacy between a regular voter and a conspiring voter. The above privacy definitions capture this by determining the exact choice group and thus the exact voter privacy for any level of conspiracy.

4.6 Integrating privacy primitives

The framework as introduced in this chapter focuses on expressing systems. In order focus on the framework, the privacy-enhancing primitives discussed in Section 3.1 were left out. Having established the framework, this section integrates these primitives. Integrations of the primitives are introduced in the same order as discussed previously: properties of cryptographic systems (homomorphic encryption, blind signatures, re-encryption), shared secret decryption and proofs.

4.6.1 Properties of cryptographic systems

The properties of cryptographic systems used for privacy operate on terms. As such, the extensions for these need only discuss those parts of the framework that deal with terms. The relevant parts are:

- the term algebra,
- term derivation $K \vdash \varphi$,
- deconstruction of terms $\varphi' \sqsubseteq \varphi$ (for readability),
- reinterpretation of terms $\rho(\varphi)$,
- the functions governing voter controllable variables of terms, $\text{vars}(v, \varphi)$ and $\text{freshvars}(v, \varphi)$.

The integration each of these primitives is discussed in the above order. Note that the definitions of substitution (Definition 2), matching (Definition 3) and readability (Definition 8) themselves do not depend on the term structure. As such, these definitions do not need to be adapted. Furthermore, all extensions are observable by default. As such, the definition of the observable function *obstr* does not change with the below extensions.

Homomorphic encryption

To model a specific homomorphic encryption system on terms, we extend the term algebra as follows.

$$\varphi ::= \dots \mid \varphi_1 \oplus \varphi_2 \mid \llbracket \varphi \rrbracket_k$$

Homomorphic encryption of a term φ with key k is denoted as $\llbracket \varphi \rrbracket_k$, while $\varphi_1 \oplus \varphi_2$ denotes the plaintext homomorphic operator. Note that due to the equivalence HE1 (see Section 3.1.1), we do not need to introduce the ciphertext operator \otimes .

Derivation of these new terms is governed by the following extension to the derivation rules.

$$\begin{array}{lll} \text{(encryption)} & K \vdash \varphi, K \vdash k & \Longrightarrow K \vdash \llbracket \varphi \rrbracket_k \\ \text{(decryption)} & K \vdash \llbracket \varphi \rrbracket_k, K \vdash k^{-1} & \Longrightarrow K \vdash \varphi \\ \text{(homomorphic operation)} & K \vdash \llbracket \varphi_1 \rrbracket_k, K \vdash \llbracket \varphi_2 \rrbracket_k & \Longrightarrow K \vdash \llbracket \varphi_1 \oplus \varphi_2 \rrbracket_k \end{array}$$

The deconstruction of a homomorphically encrypted term $\llbracket \varphi \rrbracket_k$ is equivalent to the deconstruction of regular encryptions.

$$\varphi \sqsubseteq \llbracket \varphi \rrbracket_k, \quad k^{-1} \sqsubseteq \llbracket \varphi \rrbracket_k$$

Reinterpretation of homomorphically encrypted terms is similar to reinterpretation of normally encrypted terms.

$$\rho(\llbracket \varphi \rrbracket_k) = \llbracket \rho(\varphi) \rrbracket_{\rho(k)}, \quad \text{if } K_I \vdash \varphi, k \vee K_I \vdash \llbracket \varphi \rrbracket_k, k^{-1}$$

Finally, the voter-controllable term functions are expanded as follows.

$$\begin{array}{ll} \text{vars}(v, \llbracket \varphi \rrbracket_k) = \text{vars}(v, \varphi), & \text{if } k \in \text{Keys}_v \\ \text{freshvars}(v, \llbracket \varphi \rrbracket_k) = \begin{cases} \llbracket \text{freshvars}(v, \varphi) \rrbracket_k, & \text{if } k \in \text{Keys}_v \\ \llbracket \varphi \rrbracket_k & \text{otherwise} \end{cases} \end{array}$$

Blind signatures

To encompass blind signatures, we extend the term algebra as follows.

$$\varphi ::= \dots \mid \llbracket \varphi \rrbracket_k$$

In the derivation rules, the unique property of blinding is expressed: an agent that knows the blinding key k can derive a signed, unblinded term from a signed, blinded term.

$$\begin{array}{lll} \text{(blinding)} & K \vdash \varphi, K \vdash k & \Longrightarrow K \vdash \llbracket \varphi \rrbracket_k \\ \text{(deblinding)} & K \vdash \llbracket \varphi \rrbracket_k, K \vdash k & \Longrightarrow K \vdash \varphi \\ \text{(deblind signing)} & K \vdash \{\llbracket \varphi \rrbracket_k\}_{sk(a)}, K \vdash k & \Longrightarrow K \vdash \{\varphi\}_{sk(a)} \end{array}$$

Note that blinding thus resembles encryption with a key k where $k^{-1} = k$.

The deconstruction of a blinded term results in the used key and the term itself, as follows.

$$\varphi \sqsubseteq \llbracket \varphi \rrbracket_k, \quad k \sqsubseteq \llbracket \varphi \rrbracket_k$$

Reinterpretation of blinded terms occurs if the intruder can deblind the term, or if the intruder can construct the blinded term.

$$\rho(\llbracket \varphi \rrbracket_k) = \llbracket \rho(\varphi) \rrbracket_{\rho(k)}, \quad \text{if } K_I \vdash k \wedge (K_I \vdash \varphi \vee K_I \vdash \llbracket \varphi \rrbracket_k)$$

Finally, the functions dealing with voter-controllable variables are extended for blind terms as follows:

$$\begin{aligned} \text{vars}(v, \llbracket \varphi \rrbracket_k) &= \text{vars}(v, \varphi), & \text{if } k \in \text{Keys}_v \\ \text{freshvars}(v, \llbracket \varphi \rrbracket_k) &= \begin{cases} \llbracket \text{freshvars}(v, \varphi) \rrbracket_k & \text{if } k \in \text{Keys}_v \\ \llbracket \varphi \rrbracket_k & \text{otherwise} \end{cases} \end{aligned}$$

Re-encryption

Re-encryption is an encryption scheme where the form of the ciphertext can be changed without the key. This change does not affect the encrypted plaintext, but a random factor. As such, this random factor must be modelled and distinguishable from the encrypted term in the term algebra. Thus, we extend the term algebra as follows.

$$\varphi ::= \dots \mid \langle \langle \varphi_1 \mid \varphi_2 \rangle \rangle_k$$

Here, φ_1 represents the encrypted term, and φ_2 models the manipulable part of the randomness of the encryption.

The derivation rules for re-encryption capture its properties. The randomness in the encryption is modelled as pairing of nonces. Therefore, re-encryption can only extend the randomness with nonces, and not with arbitrary terms. Note that extending the randomness does not require any key.

$$\begin{aligned} (\text{encryption.}) \quad K \vdash \varphi, K \vdash n, K \vdash k &\implies K \vdash \langle \langle \varphi \mid n \rangle \rangle_k \\ (\text{decryption}) \quad K \vdash \langle \langle \varphi_1 \mid \varphi_2 \rangle \rangle_k, K \vdash k^{-1} &\implies K \vdash \varphi_1 \\ (\text{re-encryption}) \quad K \vdash \langle \langle \varphi_1 \mid \varphi_2 \rangle \rangle_k, K \vdash n &\implies K \vdash \langle \langle \varphi_1 \mid (\varphi_2, n) \rangle \rangle_k \end{aligned}$$

The deconstruction of a re-encryptable term is as follows. Note that the randomness is *not* part of the deconstruction of the encrypted term. This is because the only effect of the random factor is to ensure a variation in the ciphertext.

$$\varphi_1 \sqsubseteq \langle \langle \varphi_1 \mid \varphi_2 \rangle \rangle_k, \quad k^{-1} \sqsubseteq \langle \langle \varphi_1 \mid \varphi_2 \rangle \rangle_k$$

Finally, the functions dealing with voter controllable variables are extended as follows.

$$\begin{aligned} \text{vars}(v, \langle \langle \varphi_1 \mid \varphi_2 \rangle \rangle_k) &= \text{vars}(v, \varphi_1) & \text{if } k \in \text{Keys}_v \\ \text{freshvars}(v, \langle \langle \varphi_1 \mid \varphi_2 \rangle \rangle_k) &= \begin{cases} \langle \langle \text{freshvars}(v, \varphi_1) \mid \varphi_2 \rangle \rangle_k & \text{if } k \in \text{Keys}_v \\ \langle \langle \varphi_1 \mid \varphi_2 \rangle \rangle_k & \text{otherwise} \end{cases} \end{aligned}$$

4.6.2 Shared secret decryption

We describe here how the framework is extended to capture shared secret decryption, in which secret key k^{-1} is *not* reconstructed (see Section 3.1.3). We briefly recapitulate

the setting. There is an encryption key k and a decryption key k^{-1} . The decryption key k^{-1} is split over n shares. We consider threshold sharing, i.e. $t \leq n$ authorities need to participate to obtain a decryption. In a t, n threshold secret sharing scheme, there are n shares of secret k . We denote the i^{th} share, (for $0 \leq i < n$), as ${}_{t,n}\text{share}_{k,i}$.

To decrypt an encrypted term $\{\varphi\}_k$ without reconstructing k^{-1} , specific decryption shares are used. These shares depend on the term to be decrypted, and we denote such a share as ${}_{t,n}\text{share}_{k,i}^{\{\varphi\}_k}$. With t of such shares, the term can be decrypted.

The term algebra is extended as follows for shared secret decryption.

$$\varphi ::= \dots \mid {}_{t,n}\text{share}_{k,i} \mid {}_{t,n}\text{share}_{k,i}^{\{\varphi\}_k}, \quad \text{for } 0 \leq i < n \text{ and } t, n \in \mathbb{N}.$$

A decryption share can be derived from a share of k and an encrypted term. With t decryption shares, an encrypted term can be decrypted. This is captured by the following derivation rules.

$$\begin{aligned} \text{(decrypt share)} \quad & K \vdash {}_{t,n}\text{share}_{k,i} \quad K \vdash \{\varphi\}_k \implies K \vdash {}_{t,n}\text{share}_{k,i}^{\{\varphi\}_k} \\ \text{(decryption)} \quad & K \vdash \{\varphi\}_k, K \vdash Z_{t,n}, |Z_{t,n}| \geq t \implies K \vdash \varphi, \\ & \text{where } Z_{t,n} \subseteq \{{}_{t,n}\text{share}_{k,i}^{\{\varphi\}_k} \mid 0 \leq i < n\}. \end{aligned}$$

An agent receiving a share cannot specify the structure of the share. Thus, a share is a term, which cannot be deconstructed. The same holds for decryption shares. Consequently, either can be freely reinterpreted. Furthermore, as shares have no internal structure, there is no need to extend the vars and freshvars functions. Thus the above extensions complete the modelling of shared secret decryption in the framework.

4.6.3 Integrating proofs

In Section 3.1.2, we discussed two types of proofs: zero knowledge proofs (ZKP) and designated verifier proofs (DVP). The precise modelling of a proof depends on what is being proven, as well as which properties the proof is to have. As such, this section serves as an illustration of how to extend the framework with ZKP and DVP proofs.

Voting systems that use mixnets may use a designated verifier zero knowledge proof (DZP) to prove that a term φ_1 is a re-encryption of a term φ_2 . An example of a voting scheme using DZP is [LBD⁺03]. As this type of proof combines both the zero knowledge property, and the designated verifier property, it is well-suited to illustrate how to model these proofs in the framework.

Designated verifier zero knowledge proofs of re-encryption

Proofs are communicated between agents. Hence they must be terms, and thus, as before, we introduce extensions at the term level. Furthermore, proofs steer decisions by agents. As such, we introduce the possibility for an agent to choose, based on a proof. This new process construct has an impact on every definition that depends on the structure of processes. These are the process substitution $\sigma(P)$, the free variables of a process $\text{fv}(P)$, the semantics rules, and the process transformation function $\Theta_i(v, P)$.

Extensions to terms. A DZP of re-encryption proves that two terms of the form $\langle\langle\varphi_1 \mid \varphi'_1\rangle\rangle_k, \langle\langle\varphi_2 \mid \varphi'_2\rangle\rangle_k$, which are encrypted with the same key, are related via extra randomness n . The proof is keyed (literally) to a designated verifier a by means of the designated verifier's public key $pk(a)$. The class of terms is extended as follows.

$$\varphi ::= \dots \mid dzp(\langle\langle\varphi_1 \mid \varphi'_1\rangle\rangle_k, \langle\langle\varphi_2 \mid \varphi'_2\rangle\rangle_k, pk(a))$$

Note that a DZP does *not* include the randomness n . This makes the zero knowledge aspect of the proof explicit: the proof does not give any information concerning n . However, to construct a DZP, this randomness is needed.

For readability, we omit the structure of the re-encryption when it is not relevant. Thus, we will write a DZP as $dzp(\varphi, \varphi', pk(a))$, for terms φ, φ' which are both re-encryptions.

The derivation rules are extended as follows for DZP's.

$$\begin{aligned} \text{(generate DZP)} \quad & K \vdash \langle\langle\varphi_1 \mid \varphi_2\rangle\rangle_k, n, pk(a) \\ & \implies K \vdash dzp(\langle\langle\varphi_1 \mid \varphi_2\rangle\rangle_k, \langle\langle\varphi_1 \mid (\varphi_2, n)\rangle\rangle_k, pk(a)) \\ \text{(fake DZP)} \quad & K \vdash \langle\langle\varphi_1 \mid \varphi'_1\rangle\rangle_k, \quad K \vdash \langle\langle\varphi_2 \mid \varphi'_2\rangle\rangle_k, \quad K \vdash sk(a) \\ & \implies K \vdash dzp(\langle\langle\varphi_1 \mid \varphi'_1\rangle\rangle_k, \langle\langle\varphi_2 \mid \varphi'_2\rangle\rangle_k, pk(a)) \\ \text{(extract terms)} \quad & K \vdash dzp(\varphi, \varphi', pk(a)) \\ & \implies K \vdash \varphi \wedge K \vdash \varphi' \wedge K \vdash pk(a) \end{aligned}$$

Given a term $\langle\langle\varphi_1 \mid \varphi'_1\rangle\rangle_k$, a nonce n , and a public key $pk(a)$, an agent can derive a DZP for agent a that proves that $\langle\langle\varphi_1 \mid (\varphi'_1, n)\rangle\rangle_k$ is a re-encryption of the original term. Any agent can generate fake DVP's with himself as designated verifier, as captured by the second derivation rule. Faking is only possible with an agent's secret key, ensuring that no other agent can create a fake DZP for an agent.

A DZP can be deconstructed as follows.

$$\begin{aligned} \varphi & \sqsubseteq dzp(\varphi, \varphi', pk(a)) \\ \varphi' & \sqsubseteq dzp(\varphi, \varphi', pk(a)) \\ sk(a) & \sqsubseteq dzp(\varphi, \varphi', pk(a)) \end{aligned}$$

Reinterpretation of a DZP depends on the reinterpretability of the terms.

$$\rho(dzp(\varphi, \varphi', k)) = dzp(\rho(\varphi), \rho(\varphi'), \rho(k))$$

The functions handling voter controllable variables are extended as follows.

$$\begin{aligned} \text{vars}(v, dzp(\varphi, \varphi', k)) &= \text{vars}(v, \varphi) \cup \text{vars}(\varphi') \\ \text{freshvars}(v, dzp(\varphi, \varphi', k)) &= \\ & \quad dzp(\text{freshvars}(v, \varphi), \text{freshvars}(v, \varphi'), \text{freshvars}(v, k)) \end{aligned}$$

Extensions to process syntax. Agents can make decisions based on a proof. Thus, we introduce a new choice construction, that of checking a proof. The class of Processes is extended as follows.

$$P ::= \dots \mid P_1 \triangleleft chkprf(dzp(\varphi, \varphi', k)) \triangleright P_2$$

The free variables of a process of the above form are defined as follows.

$$\text{fv}(P_1 \triangleleft \text{chkprf}(\text{dzp}(\varphi, \varphi', k)) \triangleright P_2) = \text{fv}(P_1) \cup \text{fv}(P_2) \cup \text{fv}(\text{dzp}(\varphi, \varphi', k))$$

The definition of substitution is also extended to the new choice construction.

$$\begin{aligned} \sigma(P_1 \triangleleft \text{chkprf}(\text{dzp}(\varphi, \varphi', k)) \triangleright P_2) = \\ \sigma(P_1) \triangleleft \text{chkprf}(\text{dzp}(\sigma(\varphi), \sigma(\varphi'), \sigma(k))) \triangleright \sigma(P_2) \end{aligned}$$

Finally, process transformation is extended to the new process structure.

$$\begin{aligned} \Theta_i(v, P_1 \triangleleft \text{chkprf}(\text{dzp}(\varphi, \varphi', k)) \triangleright P_2) = \\ \Theta_i(P_1) \triangleleft \text{chkprf}(\text{dzp}(\varphi, \varphi', k)) \triangleright \Theta_i(P_2) \end{aligned}$$

Extensions to semantics. The new process structure introduced above expresses a choice based on a proof. To make this choice, we introduce the predicate *Correct*, which, given a proof and a key, determines whether the proof seems correct with the supplied key. However, *Correct* cannot distinguish between fake DZP's and real DZP's.

$$\begin{aligned} \text{Correct}(\text{dzp}(\langle \{\varphi_1 \mid \varphi'_1\} \rangle_k, \langle \{\varphi_2 \mid \varphi'_2\} \rangle_k, pk(a)), k') = \\ \begin{cases} \text{false} & \text{if } k' = sk(a) \wedge \varphi_1 \neq \varphi_2 \\ \text{true} & \text{otherwise} \end{cases} \end{aligned}$$

Thus, all DZP's seem correct, even false ones, except when checked with the right key. This expresses the designated verifier aspect of evaluating such proofs. As a consequence, whether an agent accepts a proof depends not only on the apparent correctness of the proof, but also on whether or not the agent possesses the right key.

The semantic rules for evaluating DZP's are specified as agent level semantic rules, and are given below. Note that an agent will reject a proof if the agent does not know the necessary key, or (when he does know the necessary key) if the proof is fake.

(correct DZP)

$$\begin{aligned} (K_I, \text{knw}_v, P_1) &\xrightarrow{ev} (K'_I, \text{knw}'_v, P) \\ \text{prf} &= \text{dzp}(\langle \{\varphi_1 \mid \varphi_2\} \rangle_k, \langle \{\varphi_1 \mid \varphi'_2\} \rangle_k, pk(v)) \\ \frac{\text{knw}_v \vdash sk(v) \quad \text{knw}_v \vdash \text{prf} \quad \text{Correct}(\text{prf}, sk(v))}{(K_I, \text{knw}_v, P_1 \triangleleft \text{chkprf}(\text{prf}) \triangleright P_2) &\xrightarrow{ev} (K'_I, \text{knw}'_v, P)} \end{aligned}$$

(fake DZP)

$$\begin{aligned} (K_I, \text{knw}_v, P_2) &\xrightarrow{ev'} (K'_I, \text{knw}'_v, P') \\ \text{prf} &= \text{dzp}(\langle \{\varphi_1 \mid \varphi'_1\} \rangle_k, \langle \{\varphi_2 \mid \varphi'_2\} \rangle_k, pk(a)) \\ \frac{\text{knw}_v \vdash \text{prf} \quad \text{knw}_v \not\vdash sk(a) \vee \neg \text{Correct}(\text{prf}, sk(a))}{(K_I, \text{knw}_v, P_1 \triangleleft \text{chkprf}(\text{prf}) \triangleright P_2) &\xrightarrow{ev'} (K'_I, \text{knw}'_v, P')} \end{aligned}$$

4.6.4 Summary

This section extended the framework of Chapter 4 to capture several primitives that are used in voting systems. The extensions capture the following primitives:

- homomorphic encryption,
- blind signatures,
- re-encryption,
- shared secret decryption,
- designated verifier zero knowledge proofs.

These extensions make modelling specific voting systems easier and ensure that the models are close to the original system specification. In the rest of this chapter, we freely use the above extensions if and when relevant.

4.7 Discussions

This section discusses how the privacy definitions of the framework extend to privacy with respect to voting authorities, and how the definitions relate to the notions of receipt-freeness and coercion-resistance.

4.7.1 Privacy from voting authorities

The framework of this chapter assumes that the intruder is an outside observer. As such, the privacy measurements of the framework do not extend to privacy with respect to voting authorities. However, this is easily achieved by letting the authority acts as the intruder.

4.7.2 Receipt-freeness

Receipt-freeness, as introduced by Benaloh and Tuinstra [BT94], captures ways for a voter to reduce the privacy of how she voted. The framework captures receipt-freeness by specifying the desired class of conspiring behaviour. From the hierarchy in Figure 4.2, we conclude that conspiring behaviour of type $2c$ models the most conspiring behaviour in the system. Any knowledge possessed by the voter is shared as soon as possible with the intruder. Nevertheless, this does not preclude $2c$ -conspiracy-resistant voting systems. For example, designated verifier proofs still force privacy on the voter, as the intruder has no way to determine if a designated verifier proof forwarded by the voter is fake or not.

4.7.3 Coercion-resistance

Coercion-resistance, as introduced by Juels, Catalano and Jakobsson [JCJ05], captures ways for a voter to reduce the privacy of how she interacts with the voting

system. There is confusion in literature about the difference between the notions of coercion-resistance and receipt-freeness, and with good cause: if the voter can prove how she voted, the coercer can force her to produce this proof. Conversely, if there is no such proof, the coercer cannot force this proof. Thus, the susceptibility of a voter to be forced to vote in a specific way is equivalent to her ability to prove that she voted in that specific way. Succinctly put:

Proving ability = coercion susceptibility.

The notion of coercion-resistance as introduced by Juels et al. extends beyond reducing privacy of the vote. In their view, coercion-resistance encompasses the following:

- **receipt-freeness:** the voter proves how she voted to the intruder.
- **forced abstention:** the intruder prevents the voter from voting.
- **simulation attacks:** the voter gives her private keys to the intruder, who votes in her stead.
- **forced random voting:** the intruder forces the voter to vote for a random entry in a list of encrypted candidates. Note that the intruder does not need to know which candidate is chosen, as long as it is a random choice. This attack (if applied to many voters) forces a more uniform distribution of votes instead of the actual distribution of votes. Such a strategy benefits candidates with less than average votes at the cost of harming candidates with more than average votes.

Receipt-freeness is captured by the framework, as explained above.

Forced abstention. To capture forced abstention, we extend the range of γ with an “abstention” candidate \perp . This candidate embodies all votes, that do not affect the result (we do not distinguish between abstention and malformed voting). The extended set is referred to as \mathcal{C}_\perp . Using these extensions, the intruder can force voter v with conspiring behaviour i to abstain, iff

$$cg_v^i(\mathcal{VS}, \gamma) = \{\perp\}.$$

Simulation attack. In a simulation attack, the intruder casts a vote himself. Simulation attacks are resisted if the intruder cannot tell if the vote he cast affects the result or not. While the type of voter behaviour necessary for a simulation attack is modelled by the framework (type 2 behaviour), intruder voting is not. Therefore, we extend the domain of γ to include the intruder int . The extended set is referred to as \mathcal{V}_{int} . The intruder’s choice group cg_{int} then captures the intruder’s unsureness about his own vote. Using this, a system is simulation-attack-resistant if the intruder cannot tell whether his vote is counted or not, i.e.

$$\{\perp, \gamma(int)\} \subseteq cg_{int}(\mathcal{VS}, \gamma).$$

Forced random voting. In forced random voting attacks, the intruder forces the voter to vote randomly. This means that whenever the voter process can make a choice, either conditionally or non-deterministically, the intruder instructs the voter how to proceed. This can be expressed in terms of the framework by rewriting every process P that denotes a choice. Let Δ_{rnd} be a process transformation function for forcing choices. Then we have, for any process P such that $P = P_1 + P_2$ and for any process P such that $P = P_1 \triangleleft \varphi_1 = \varphi_2 \triangleright P_2$,

$$\Delta_{rnd}(P) = ir(\mathbf{var}).\Delta_{rnd}(P_1) \triangleleft \mathbf{var} = true \triangleright \Delta_{rnd}(P_2),$$

where \mathbf{var} is a fresh variable and $true$ is a constant term $\in Terms$.

Coercion resistance. Thus, coercion attacks can be captured in the framework as well. However, we keep these latter attacks separate, as these attacks are fundamentally different from the privacy-reducing conspiratory model.

4.8 Conclusions

This chapter investigated related work on formalising privacy in voting and found that a quantified approach to privacy was missing in literature. To this end, a formal framework for modelling voting systems was introduced, in which voter-privacy can be quantified. Furthermore, various models of voter conspiracy were discussed and formalised in the framework. Finally, we analysed receipt-freeness and coercion-resistance on the basis of the newly established formalisation of voter-privacy.

In the next chapter, we investigate how to use the framework to determine the voter-privacy of systems such as those surveyed in Chapter 3.

5

Application

This chapter illustrates how apply the concepts introduced in Chapter 4. The purpose of this chapter is to serve as an example of how these concepts can be applied to study privacy in voting systems. To this end, we study three cases: an application to a voting scheme (FOO92) that relies on blind signatures, an application to a voting system (Prêt à Voter) that relies on physical ballots and using a quantified approach to understand the relationship between privacy and verifiability in a voting system (3BS).

Extended notation To ease readability, we extend the notation as follows. For $D = \{d1, \dots, dn\}$, and processes P_{d1}, \dots, P_{dn} , we denote the non-deterministic choice between any $P_{di}, i \in D$ as $\sum_{d \in D} P_d$. With abuse of notation,

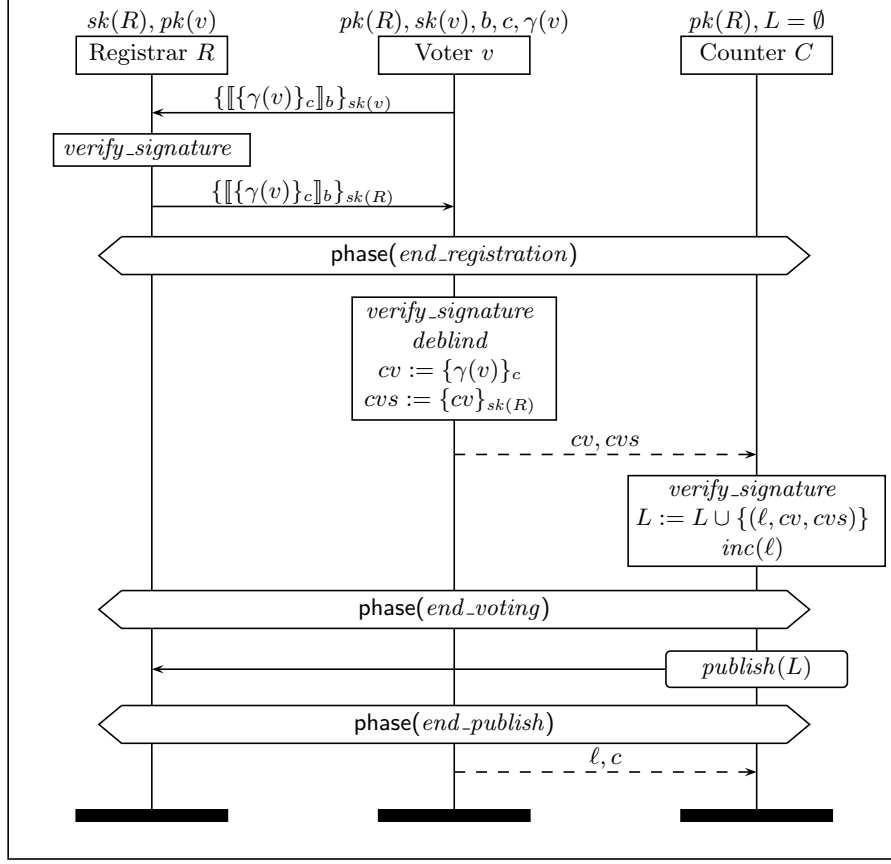
$$\sum_{d \in D} P_d = P_{d1} + \dots + P_{dn}.$$

5.1 Determining the privacy of FOO92

In this section, we apply the framework to the FOO92 scheme [FOO92]. In this scheme, there is one registrar R , who verifies eligibility of voters, and one counter C , who collects and counts the votes. There are three main phases in the scheme: the registration phase, in which encrypted votes are marked as admissible; the voting phase, in which encrypted votes are cast via an anonymous channel; and the opening phase, in which the encryption keys of the votes are communicated. The use of phases is crucial for FOO92. If phase synchronisations are omitted, a voter would register and then immediately send her committed vote to the counter over an anonymous channel, without any other voter registering in between. As no other voter could submit this committed vote, the intruder can link it to the voter (as first detailed in [KR05]). As a consequence, it is required that the registration phase may not overlap with the voting phase. A similar reasoning prohibits overlaps between the voting phase and the opening phase.

5.1.1 FOO92 description

In our modelling, we distinguish three phases in the FOO92 scheme (see Figure 5.1): a registration phase, a voting phase, and an opening phase. The registration phase

Figure 5.1: The *FOO92* scheme

begins when the voter sends her signed, blinded, committed vote to the registrar. The registrar signs the blinded and committed vote if the voter is an eligible voter, and sends this to the voter. This ends the registration phase.

In the voting phase, the voter undoes the blinding and sends the committed, registrar-signed vote via an anonymous channel (indicated by the dashed arrow in Figure 5.1) to the counter. This ends the voting phase.

In the opening phase, the counter first publishes the list L of all received votes. Then, each voter sends the key she used to the counter, together with the index ℓ of her vote in L . The counter can so open up all entries in L , and calculate the result.

5.1.2 Modelling FOO92 in the framework

The commitments in FOO92 are modelled as encryption using a nonce $c \in \text{Nonces}$. In the framework, the publishing of the final result serves to limit the possible choice functions. However, as we can see in Figure 5.1, the publication does not carry new information for the intruder. As such, we choose to omit this action in modelling the

voter.

The various phases of FOO92 are ended by phase events, which use the descriptive labels $end_registration, end_voting \in \mathbb{N}$. We introduce an auxiliary publish phase (with label $end_publish \in \mathbb{N}$), during which the list of received, committed votes is published. We denote the generic initial knowledge (the set of public keys of all agents) as knw_0 . This includes the public keys of the registrar R , the counter C and every voter $v \in \mathcal{V}$.

We omit the specific index ℓ in the modelling, as it is not needed. Then, publication of L is modelled as a public communication of L from counter C to registrar R . To ensure that all voters synchronise on this publication, we introduce a publication phase. The only event in the publication phase is the publication of L .

Voter template

Every voter behaves the same, but uses different keys for signing, blinding and commits. We describe a template $ProcV$ that describes all voter processes. This template provides a generic voter process, generalised over voter identity id , the signing key skv , the blinding key b and the commit key c .

The voter process is modelled as follows for voter v with private key $sk(v)$, blinding factor b and commitment key c (following Figure 5.1). First, voter v sends her signed, committed and blinded vote $\{\llbracket \{vc\}_c \rrbracket_b\}_{sk(v)}$ to the registrar R (recall that vc is a variable that captures the voter's vote $\gamma(v)$ when the scheme is instantiated with a choice function γ). Next, she receives her committed and blinded vote back, signed by the registrar with his private key $sk(R)$. This completes the registration phase. In the voting phase, she unblinds her vote to obtain a registrar-signed vote $\{\{vc\}_c\}_{sk(R)}$. This she sends, accompanied by the unsigned version, over an anonymous channel to the counter C . That completes the voting phase. In the publication phase, she need not take any action. After the publication phase, she sends the used commitment key c over an anonymous channel to C .

Note that this abstracts away from specific indexes ℓ in the list of received, committed votes L from Figure 5.1. An index is not private information. They do not need to be modelled to capture privacy, so we omit them. This only affects the final voter communication: she now communicates c instead of ℓ, c . Abstracting in this fashion implies that counter C must try each received key with all received votes to establish the result.

The full voter template is captured by $ProcV$ with variables $skv, b, comn, id$.

$$\begin{aligned} ProcV(sk, b, c, id) = & s(id, R, \{\llbracket \{vc\}_c \rrbracket_b\}_{sk}) \cdot \\ & r(R, id, \{\llbracket \{vc\}_c \rrbracket_b\}_{sk(R)}) \cdot \\ & ph(end_registration) \cdot \\ & as(id, C, (\{vc\}_c, \{\{vc\}_c\}_{sk(R)})) \cdot \\ & ph(end_voting) \cdot \\ & ph(end_publish) \cdot \\ & as(id, C, c) \cdot \delta \end{aligned}$$

Note that the voter's choice (denoted by vc) is left unspecified.

Given a specific signing key $sk(v)$, blinding key b and commit key c for a specific voter v , the voter's initial state is then modelled as

$$(\{sk(v), b, c\} \cup knw_0, ProcV(sk(v), b, c, v)).$$

Registrar process

The initial knowledge of the registrar consists of its private key and knw_0 . The registrar is modelled as a process $ProcR$, as follows.

During the registration phase, the registrar R accepts signed messages from every voter. Upon receipt of a signed, blinded message $\{\varphi\}_{sk(v)}$ from voter v , the registrar countersigns the message $\{\varphi\}_{sk(R)}$ and returns this countersigned message. Note that the registrar cannot undo the blinding of the message. As such, the registrar does not know what he is signing. Hence, each voter's blinded, committed vote is received by the registrar in a fresh variable **blindcomvote**. The registrar will countersign any voter's vote, which we denote by quantifying the receive and corresponding send events over $\sum_{v \in \mathcal{V}}$ (as introduced in Section 4.6).

The registrar can move to end the registration phase with a $ph(end_registration)$ event at any moment. After this, the scheme moves to the voting phase, during which the registrar has no tasks (except for ending the phase, in concert with all other authorities). This is followed by the publication phase. The only event in this phase is the communication of the list of registrar-signed, committed votes L from the counter to the registrar. After this, the phase is ended, which signifies the end of the registrar's activities.

Due to the way we model the registrar below, he will accept multiple signed votes from a single voter for registration. However, this is not a problem, as the voter processes (modelled by the $ProcV$ template) do not send in multiple votes for registration. As we are only considering voter processes as defined by $ProcV$, no voter tries to register multiple votes in our setting. As such, we omit the check for duplicate registration.

$$ProcR = \sum_{v \in \mathcal{V}} r(v, R, \{\text{blindcomvote}\}_{sk(v)}) \cdot s(R, v, \{\text{blindcomvote}\}_{sk(R)}) \cdot ProcR \\ + ph(end_registration) \cdot ph(end_voting) \cdot r(C, R, L) \cdot ph(end_publish) \cdot \delta$$

Counter process

The counter is modelled as a process $ProcC$. The counter operates in three phases: vote collection (modelled by process $VotCol$), key collection (modelled by process $KeyCol$) and publication (Pub).

The counter only begins to become active after the registration phase. Then, the counter starts collecting committed votes using the process $VotCol$. These votes are sent over an anonymous channel to the counter. The committed votes are received in a variable **comvote**, and only accepted if signed by the registrar. If accepted, the vote is added to list L .

At any time, the voting phase can end. The next phase is the publication phase, during which the counter publishes the full list of received votes by sending L to the registrar.

After this, the publication phase ends and the counter starts collecting keys using the process *KeyCol*. Keys are received over an anonymous channel into a variable k , and are stored in list KL . Ending this phase does not require synchronisation with any other agent, thus, at any moment, the counter can halt (δ).

$$\begin{aligned}
ProcC &= ph(end_registration) . VotCol(emptylist) \\
VotCol(L) &= ar(C, (comvote, \{comvote\}_{sk(R)})) . \\
&\quad VotCol((L, (comvote, \{comvote\}_{sk(R)}))) \\
&\quad + ph(end_voting) . Pub(L) \\
Pub(L) &= s(C, R, L) . ph(end_publish) . KeyCol(L, emptylist) \\
KeyCol(L, KL) &= ar(C, k) . KeyCol(L, (KL, k)) + \delta
\end{aligned}$$

5.1.3 Privacy of FOO92

We claim that FOO92 has a non-trivial choice group (i.e. a choice group of size greater than one), but is not conspiracy-resistant. In other words, FOO92 offers vote privacy, but is not resistant to coercion attacks. In order to prove this, we first prove that in any non-trivial setting (i.e. any election with the possibility of a non-unanimous result), there exists a choice function γ_1 such that the choice group of FOO92 and γ_1 contains more than one element.

Lemma 2 (privacy of FOO92). *Suppose $|\mathcal{V}| > 1$ and $|\mathcal{C}| > 1$. Then, for any choice function γ_1 , such that there exist voters $va, vb \in \mathcal{V}$ such that $\gamma_1(va) \neq \gamma_1(vb)$, we have $|cg(FOO, \gamma_1)| > 1$.*

Proof. According to Definition 14, the choice group of FOO92 for a given choice function γ_1 , $cg(FOO, \gamma_1)$, is defined as $\{\gamma_2 \mid \gamma_1 \simeq_{FOO} \gamma_2\}$. This similarity between choice functions is defined (in Definition 13) as follows.

$$\begin{aligned}
\forall t \in Tr(FOO^{\gamma_1}): \exists t' \in Tr(FOO^{\gamma_2}): t \sim t' \quad \wedge \\
\forall t' \in Tr(FOO^{\gamma_2}): \exists t \in Tr(FOO^{\gamma_1}): t' \sim t.
\end{aligned}$$

To prove that such choice functions γ_1 such that $|cg(FOO, \gamma_1)| > 1$ exist, we take such a choice function γ_1 , and a trace $t_a \in Tr(FOO^{\gamma_1})$ and show that there exists a choice function $\gamma_2 \neq \gamma_1$ such that there exists a trace $t_b \in Tr(FOO^{\gamma_2})$ for which $t_a \sim t_b$.

Let γ_1 be any non-trivial choice function, i.e. any choice function for which there are at least $va, vb \in \mathcal{V}$ and $ca, cb \in \mathcal{C}$, $va \neq vb$ and $ca \neq cb$ (possible due to $|\mathcal{V}| > 1, |\mathcal{C}| > 1$), such that $\gamma_1(va) = ca$ and $\gamma_1(vb) = cb$. Consider the choice function γ_2 such that $\gamma_2(va) = \gamma_1(vb)$, $\gamma_2(vb) = \gamma_1(va)$ and $\forall v \in \mathcal{V} \setminus \{va, vb\}: \gamma_2(v) = \gamma_1(v)$.

We now examine traces $t_a \in Tr(FOO^{\gamma_1})$ and $t_b \in Tr(FOO^{\gamma_2})$ such that all agents execute their events in exactly the same order in t_a and t_b , except that after the first phase event, vb executes in t_b whenever va executes in t_a , and vice versa (i.e. their executions are swapped from $ph(end_registration)$ onwards). Note that, for every trace $t_a \in Tr(FOO^{\gamma_1})$, there exists such a trace $t_b \in Tr(FOO^{\gamma_2})$ and vice versa. Furthermore, note that after the first phase event, the voters only engage in anonymous communication.

We will prove that $t_a \sim t_b$, and thus $\gamma_2 \simeq_{FOO} \gamma_1$. This implies that $\gamma_2 \in cg(FOO, \gamma_1)$ and thus that $|cg(FOO, \gamma_1)| \geq 2$.

First, observe that for all events by agents other than va, vb , which are not communication events with va, vb , t_a and t_b are exactly the same. Furthermore, we only focus on the send/receive events of agents va, vb . If these can be reinterpreted, so can the corresponding receive/send events by other agents.

In trace t_a , we first focus on the send/receive events of voter va . Suppose va was instantiated as $ProcV(sk(va), bva, cva, va)$. The intruder has the following view of the events in t_a and the events that occur at the same index in t_b (labelled for convenience):

	$obstr(t_a)$	$obstr(t_b)$
	\vdots	\vdots
1.	$s(va, R, \{\llbracket \{ca\}_{cva} \rrbracket_{bva}\}_{sk(va)})$	$s(va, R, \{\llbracket \{cb\}_{cva} \rrbracket_{bva}\}_{sk(vb)})$
	\vdots	\vdots
2.	$r(R, va, \{\llbracket \{ca\}_{cva} \rrbracket_{bva}\}_{sk(R)})$	$r(R, va, \{\llbracket \{cb\}_{cva} \rrbracket_{bva}\}_{sk(R)})$
	\vdots	\vdots
	$ph(end_registration)$	\vdots
	\vdots	\vdots
3.	$as(C, (\{ca\}_{cva}, \{\{ca\}_{cva}\}_{sk(R)}))$	$as(C, (\{ca\}_{cvb}, \{\{ca\}_{cvb}\}_{sk(R)}))$
	\vdots	\vdots
4.	$as(C, cva)$	$as(C, cvb)$
	\vdots	\vdots

Note that a trace does not need to describe a full execution of the scheme. It can consist of any number of labels. Thus, t_a may not contain all of these events. But as we pick t_b to match the order of execution of events in t_a , it also lacks the appropriate events and thus is similar.

Below we describe a reinterpretation ρ_a such that $\rho_a(t_a) = t_b$ (which does not need to reinterpret events 1 and 2). Note that for all agents a , $pk(a) \in K_I$, and that $cva \in K_I$ due to event 4 above. However, this is not true for the blinding key: $bva \notin K_I$. This is because only va knows bva , and she does not communicate this key. Therefore, the intruder cannot undo the blinding in events 1 and 2. Thus, he is not aware of the inner contents of the messages of events 1 and 2. With this in mind, we have the following reinterpretation.

$$\begin{aligned} \rho_a((\{ca\}_{cva}, \{\{ca\}_{cva}\}_{sk(R)})) &= (\{ca\}_{cvb}, \{\{ca\}_{cvb}\}_{sk(R)}) && \text{event 3} \\ \rho(cva) &= cvb && \text{event 3,4} \end{aligned}$$

By treating the events of vb in trace t_a similar as above, we extend ρ_a as follows.

$$\begin{aligned} \rho_a((\{cb\}_{cvb}, \{\{cb\}_{cvb}\}_{sk(R)})) &= (\{cb\}_{cva}, \{\{cb\}_{cva}\}_{sk(R)}) \\ \rho(cvb) &= cva \end{aligned}$$

Thus, we have $\rho_a(t_a) = t_b$. In a similar fashion, we can construct ρ_b such that $\rho_b(t_b) = t_a$. As we did not specify any restrictions on t_a , this holds for any $t_a \in Tr(FOO^{\gamma_1})$, which proves $\forall t \in Tr(FOO^{\gamma_1}): \exists t' \in Tr(FOO^{\gamma_2}): t \sim t'$. The proof for the other conjunct, $\forall t \in Tr(FOO^{\gamma_1}): \exists t' \in Tr(FOO^{\gamma_2}): t \sim t'$, is equivalent. \square

Given that FOO92 has non-trivial choice groups, we can determine the impact of voter conspiracy. We claim that FOO is not conspiracy-resistant.

Lemma 3 (non-conspiracy-resistance of FOO92). *For any non-trivial setting (i.e. $|\mathcal{V}| > 1, |\mathcal{C}| > 1$), the following holds:*

1. *FOO is not conspiracy resistant for conspiring behaviour of type 1.*
2. *FOO is not conspiracy resistant for conspiring behaviour of type 2.*

Proof.

Type 1 conspiracy. Conspiracy-resistance of FOO with respect to conspiring behaviour of type 1 is defined as (Definition 18):

$$\forall v \in \mathcal{V}, \gamma \in \mathcal{V} \rightarrow \mathcal{C}: cg_v(\Theta_1(v, FOO), \gamma) = cg_v(FOO, \gamma).$$

This compares the regular FOO scheme with a version where the process P_v of one voter v is transformed according to $\Theta_1(v, P_v)$. This process transformation prepends an $is(v, knw_v)$ before any δ process. This leads to the following transformation for a conspiring voter v with blinding key b , commit key c and signing key $sk(v)$ (the differences with the regular voting process are emphasised **in this font**):

$$\begin{aligned} \Theta_1(v, ProcV(sk(v), b, c, v)) = & s(v, R, \{\llbracket \{vc\}_c \rrbracket_b\}_{sk(v)}) \cdot r(R, v, \{\llbracket \{vc\}_c \rrbracket_b\}_{sk(R)}) \cdot \\ & ph(end_registration) \cdot \\ & as(v, C, (\{vc\}_c, \{\{vc\}_c\}_{sk(R)})) \cdot \\ & ph(end_voting) \cdot \\ & ph(end_publish) \cdot \\ & as(v, C, c) \cdot \mathbf{is}(v, \mathbf{knw}_v) \cdot \delta \end{aligned}$$

By Lemma 2, there exists a choice function γ_1 such that $|cg(FOO, \gamma_1)| > 1$ (i.e. a non-unanimous vote). Then, there are at least $\gamma_1, \gamma_2 \in cg(FOO, \gamma_1)$, such that $\gamma_1 \neq \gamma_2$. Thus, there is at least one voter va , for which $\gamma_1(va) \neq \gamma_2(va)$. Consequently, $\{\gamma_1(va), \gamma_2(va)\} \subseteq cg_{va}(FOO, \gamma_1)$.

We prove that $\gamma_2(va) \notin cg_{va}(\Theta_1(va, FOO), \gamma_1)$. Let $ca = \gamma_1(va)$ and $cb = \gamma_2(va)$. As for Lemma 2, we consider a trace t_a , but in this case $t_a \in Tr(\Theta_1(va, FOO)^{\gamma_1})$.

$$\begin{array}{c} \hline obstr(t_a) \\ \hline \vdots \\ 1. \quad s(va, R, \{\llbracket \{ca\}_{cva} \rrbracket_{bva}\}_{sk(va)}) \\ \vdots \\ 2. \quad r(R, va, \{\llbracket \{ca\}_{cva} \rrbracket_{bva}\}_{sk(R)}) \\ \vdots \\ 3. \quad as(C, (\{ca\}_{cva}, \{\{ca\}_{cva}\}_{sk(R)})) \\ \vdots \\ 4. \quad as(C, cva) \\ \vdots \\ 5. \quad \mathbf{is}(va, \mathbf{knw}_{va}) \\ \vdots \end{array}$$

Due to event 4 and the conspiring voter's event 5, we have $cva, bva \in K_I^{t_a}$. As such, the intruder *can* open the blinding in events 1 and 2. Thus, we have that any reinterpretation ρ of event 1 obeys the following.

$$\begin{aligned} & \rho(s(va, R, \{\llbracket \{ca\}_{cva} \rrbracket_{bva} \rrbracket_{sk(va)}\})) \\ = & s(va, R, \rho(\{\llbracket \{ca\}_{cva} \rrbracket_{bva} \rrbracket_{sk(va)}\})) \\ = & s(va, R, \{\llbracket \{ca\}_{\rho(cva)} \rrbracket_{\rho(bva)} \rrbracket_{sk(va)}\}) \end{aligned}$$

As the intruder possesses *bva* and *cva*, he can open the blinding and see *ca* in this message. Since $ca \in \mathcal{C}$, the intruder cannot reinterpret *ca* in message 1 at all (see Definition 11). Hence, there is no reinterpretation with any trace in which *va* votes for *cb*.

Type 2 conspiracy. Lack of conspiracy-resistance of FOO92 for conspiring behaviour of type 2 follows directly from the lack of conspiracy-resistance for conspiring behaviour of type 1. \square

Whereas a type 2 conspiring voter shares her knowledge immediately, a type 1 conspiring voter ends by sharing her knowledge. In the case of a type 2 conspiring voter, $cva \in K_I^{t_a}$ for any trace with at least one event of voter *va*. As such, no occurrence of message 3 in a trace can be reinterpreted. This contrasts with type 1 conspiracy, where message 3 can be reinterpreted as long as neither message 4 nor message 5 occur in the trace (this can happen for traces of partial execution).

5.1.4 Discussion

While FOO92 has non-trivial choice groups, it cannot offer conspiracy-resistance. The application of the framework to FOO92 used an extension (blind signatures) and resulted in a proof of a positive result (Lemma 2) and a proof of a negative result (Lemma 3). Lemma 2 proves that FOO92 offers privacy, while Lemma 3 proves that FOO92 does not enforce privacy. Voters can nullify their privacy in FOO92. The problem is that a voter can reveal her blinding factor *bva*, which uniquely identifies the voter *and* is the only secret keeping her vote private.

This prompts the question of how to introduce conspiracy-resistance in FOO92. There are two messages preventing conspiracy-resistance in FOO92. In message 1, the vote is kept private by blinding with a blinding factor known only to the voter. The contents of this message can thus be revealed by the voter, meaning that it can no longer be reinterpreted. Furthermore, message 3 sends the vote of the voter encrypted with a key known only by her (at that specific point in the trace). A voter who shares this key in a timely fashion with the intruder (i.e. any time before message 4) prevents reinterpretation of this message, even if message 4 does not occur in the trace.

One possible way to mitigate these two problems is to use a trusted computation device, as proposed by Burmester et al. [MBC01] and by Baudron et al. [BFP⁺01]. The voter could send an encrypted vote to the trusted device. The trusted device would then re-encrypt and blind the vote. In this fashion, the voter would not possess secrets to share. However, the trusted device should prove to the voter that it has performed its operations correctly (e.g. along the lines of the re-encryption proofs

of [HS00]. This introduces further communication between trusted device and verifier. To ensure that none of these additional communications can be leveraged by the voter to reduce privacy, any proposed solution in this direction should be fully specified and verified using the framework.

The above application of the framework to FOO92 illustrates how to use the framework to determine privacy of a voting scheme. In the next section, we illustrate ways to accounts for physical and procedural aspects of voting systems in such an analysis. To this end, we apply the framework to Prêt à Voter, an end-to-end verifiable voting system.

5.2 Voter-controlled privacy of Prêt à Voter

The Prêt à Voter voting system [Rya05], PaV for short, was one of the first end-to-end verifiable systems (cf. Chapter 3). Since then, the system has been revised and extended. Currently, the name covers a whole family of voting systems, including variants using decryption mixes [Rya05, CRS05], a variant using re-encryption mixes [RS06], a variant accounting for Single Transferable Votes [Hea07], and other extensions [XSH⁺07, LR08b, Rya08]. In this section, we focus on establishing the voter-controlled privacy of the main concepts of PaV. As the voter interaction has not changed much (in fact, conformance with existing voter experience was a main design goal of PaV), we use the version of Chaum, Ryan and Schneider [CRS05] as a basis for this section.

Note that PaV is a voting system designed for paper ballots, marked in voting booths. The security of the system partly hinges on physical security measures, such as voter isolation (in the voting booth), and hard-to-duplicate messages (which are printed on official paper). In modelling PaV, we translate these physical measures into concepts that our modelling language can handle – i.e. cryptographic operations on messages. In this fashion, modelling PaV serves as an example of how any voting system which uses similar physical properties can be captured in the framework.

PaV incorporates various procedures that ensure that the involved voting authorities cannot learn how the voter voted, while providing verifiability. As the standard application of the framework focuses on an outside intruder, we will not incorporate these procedures in our model.

The purpose of the analysis below is twofold. On the one hand, it serves to illustrate how paper ballots and procedural steps may be dealt with in an analysis using the framework. This provides the essential underpinnings for application of the framework to end-to-end verifiable systems and to paper ballot systems. On the other hand, the analysis serves to determine the control a voter has in PaV over her privacy, with respect to an outsider. For this latter reason, the analysis is focused on voter interaction. Understanding the source of this control will aid in adapting PaV for remote voting. Note that the voter interacts with paper ballots and with PaV procedures, so this focus does not detract from the first purpose.

5.2.1 Description of PaV

We distinguish three roles in PaV: voters, who vote; registrar(s), who supply voters with their ballots; and counter(s) (dubbed tellers in the PaV literature), who (jointly) create the ballots and count the votes.

The actions in PaV occur in three distinct phases: setup, vote-casting and counting. Below, each phase is discussed in order.

Setup phase

In the setup phase, the counters create the ballots. Ballots have a traditional appearance: a left-hand side listing candidates, and a right-hand side where the voter can mark her preference. To create a ballot, the counters use a given base ordering of the candidates. They jointly permute this order (in [CRS05], this is a cyclic shifted order). This permuted order constitutes the left-hand side of the ballot. The right-hand side has space to mark the voter's preference(s). At the bottom of the right column, below the row containing the last candidate, there is a string. This string encodes the candidate ordering and is called an onion. Figure 5.2 is an example of a (filled in) PaV-ballot that was derived from the base order “North, South, East, West” by cyclically shifting the order once.

West	
North	
South	
East	✓
	3ei82Lk

Figure 5.2: A PaV-ballot voting for East

The counters create more ballots than voters, and seal each ballot in an envelope. The sealed envelopes are given to the registrars. Thus, the registrar is not aware of the contents of any ballot.

Vote-casting phase

In the vote-casting phase, voters cast votes as follows. The voter arrives at the election office. She proves her identity to the registrar, and selects a sealed envelope from the available choice of sealed envelopes. She takes the envelope with her into the voting booth and opens the envelope. Inside the envelope, she finds a ballot resembling the one shown in Figure 5.2.

The voter places a mark on the ballot in the right column, next to the candidate she prefers. Then, the two columns are separated and the left-hand side is destroyed. This can be done inside the booth to ensure privacy of the ballot, or it can be done in front of the election officials (without showing the column) to ensure correct execution of this step.

Outside the booth, the voter receives a certified copy of the right-hand side column, and casts her vote by submitting the original column. Submitting the column can be done in different ways, for example by dropping the column into a ballot box, by feeding the strip to a device (as in [CRS05]), or by handing the strip to the registrar. Even in the latter case, the registrar does *not* learn how she voted. For example, in Figure 5.2, the right-hand column by itself does not reveal that this is a vote for “East”.

Counting phase

In the counting phase, the registrar ensures that the cast votes are received by the counters. To determine for which candidate a vote counts, the counters decrypt the onion on the vote via a mix network. This mix network is verifiable using randomised partial checking [JJR02]. Each Mixer uses two mixes to mix the cast votes, a left mix and a right mix (see Figure 5.3). The results of each mix are published. Verifiability of the entire mixing process is achieved by revealing for each vote in the intermediate mixed set either where it originated from (revealing the link to the input), or where it ended up (revealing the link to the output). The choice of which link to reveal is made by another agent, after the Mixer has mixed the votes and published the intermediate and final result. As the Mixer cannot predict which links are to be revealed, he has no choice but to correctly execute the mixing. The mixing process is shown in Figure 5.3 for three ballots.

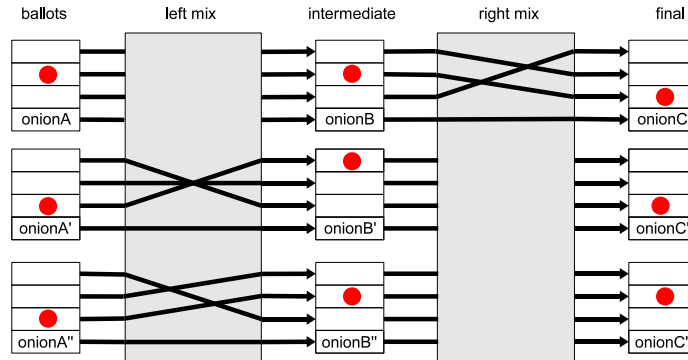


Figure 5.3: Randomised partial auditing of one Mixer

The ballots are normally also reordered, i.e. the first ballot in the input could be linked to the third ballot in the intermediate result, and the second ballot in the output. As this would only complicate the graphical representation, this shuffling has been omitted in Figure 5.3.

The ballot, as cast by the voter, is published by the registrar. Although the decryption linking ballot and candidate is protected by a mix network, this still poses a privacy problem. If the vote is published immediately after the vote was cast, then the intruder can link the ballot to the voter. On the other hand, if the registrar publishes all received votes in one go, a voter can predict one ballot (the one she cast) and thus prove *that* she voted (though not for whom). In either case, it is clear that the voter cast a vote.

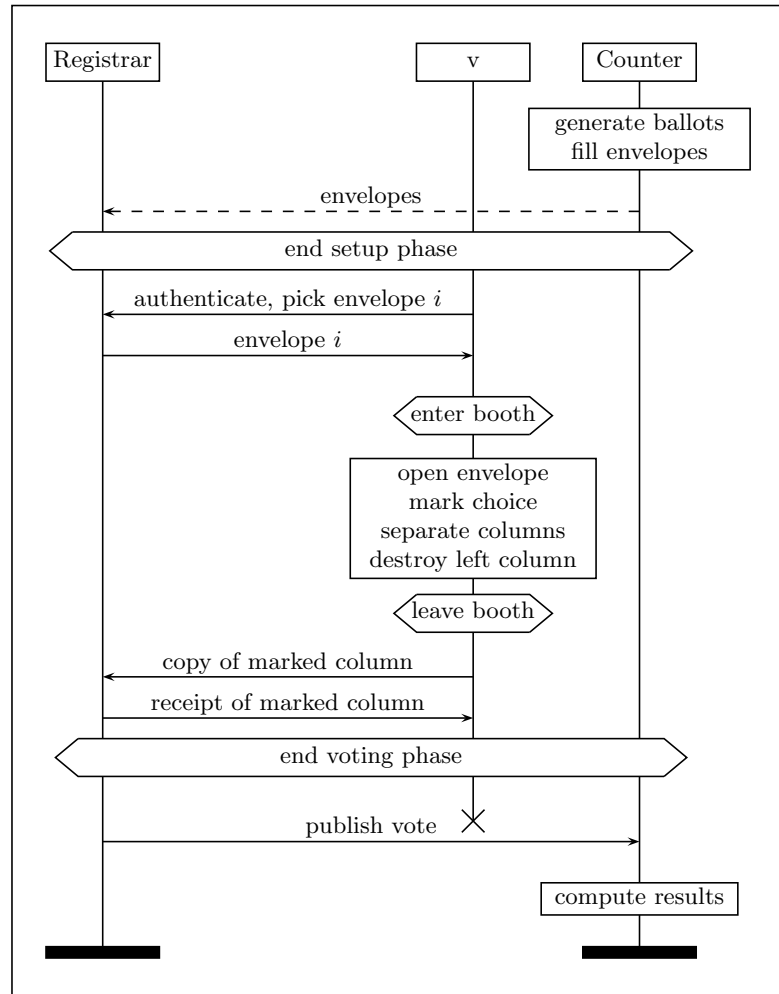


Figure 5.4: Elections with PaV

Overview of PaV elections

Figure 5.4 represents the entire PaV voting process, in the setting where the left-hand column is destroyed inside the voting booth and the right-hand column is handed to the registrar to cast the vote. In Figure 5.4, there is only one instance of the registrar and the counter. As the standard application of the framework measures privacy with respect to an outside intruder, there is no need to have multiple registrars or authorities. Thus we focus on the interaction from the voter.

5.2.2 Modelling PaV

This section describes how PaV (as shown in Figure 5.4) is modelled in the framework. The version of PaV shown in Figure 5.4 relies on several physical properties to attain

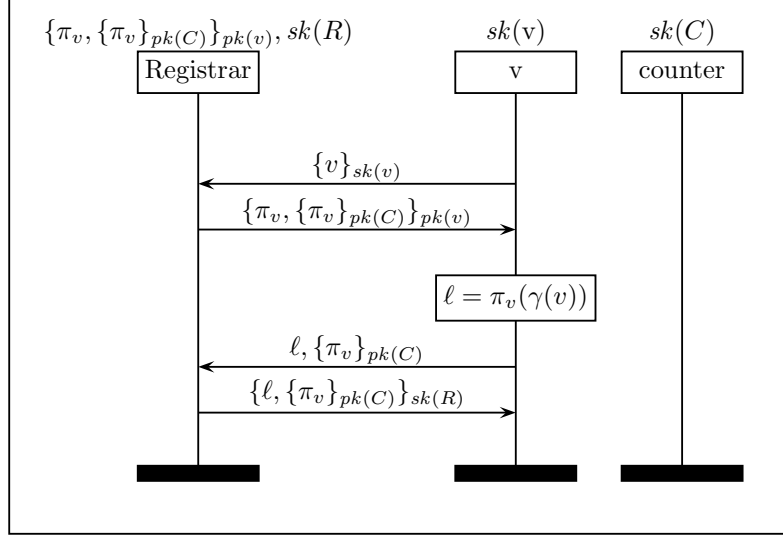


Figure 5.5: Model of voter interaction in PaV

security. Specifically, the voter's privacy is protected by the following physical and technical measures:

- Authentication proves that the voter is allowed to vote.
This is modelled using digital signatures to prove identity.
- Envelopes hide the candidate order from everyone but the voter.
This is modelled by encrypting the candidate order and the onion with the voter's public key.
- The voter chooses one envelope. No one but the voter can distinguish that envelope from other envelopes.
- A voting booth provides a private environment in which the envelopes are opened. No one can observe the actions inside the voting booth.
- The ballot lists the candidates in a random order (a random permutation in [Rya05], a cyclically shifted order in [RS06]).

As we focus on privacy with respect to an outside intruder, we will omit the measures that ensure privacy against insiders. In particular, choosing an envelope is not needed. Furthermore, as we only examine voter interaction, and as there is no interaction in the voting booth, the booth is also omitted from the model. This leads to modelling the actions in PaV as depicted in Figure 5.5, which incorporates the following:

- The permutation of the candidate order is modelled as π_v , a bijection of type $\mathcal{C} \rightarrow \{1, \dots, |\mathcal{C}|\}$. Since we do not model the voter choosing an envelope, we assume that the assignment of permutations to orders is predetermined by the system. For ease of notation, we denote the candidate order resulting from

permutation π_v as π_v as well. The index of the voter's candidate $\gamma(v)$ is denoted as $\pi_v(\gamma(v))$.

- The onion on a ballot is modelled as an encryption of the permutation with the counter's public key, i.e. $\{\pi_v\}_{pk(C)}$.

As for FOO92, we denote the generic initial knowledge (the set containing the public keys of all agents) as knw_0 . This includes the public keys of the registrar R , the counter C and every voter $v \in \mathcal{V}$.

Voter template

As in the analysis of FOO above, we have every voter executing the same actions, but with different keys and a different choice. Thus, we model these actions as a template $ProcV$, which generalises over the voter's identity and her secret key $sk(v)$. The voter's choice is left unspecified as vc . The voter template is defined as follows.

$$\begin{aligned} ProcV(id, skv) = & s(id, R, (\{id\}_{skv})) \cdot \\ & r(R, id, \{\pi_{id}, \text{onion}\}_{pk(id)}) \cdot \\ & s(id, R, (\pi_{id}(vc), \text{onion})) \cdot \\ & r(R, id, \{\pi_{id}(vc), \text{onion}\}_{sk(R)}) \cdot \delta \end{aligned}$$

A specific voter v with key pair $sk(v), pk(v)$ is then modelled as

$$(\{id, sk(v), pk(v)\} \cup knw_0, ProcV(v, sk(v))).$$

Registrar process

As discussed above, the analysis involves only one registrar. The initial knowledge of the registrar consists of its private key, the set of assigned envelopes and generic initial knowledge knw_0 . We will denote the envelope assigned to voter v as $envelope_v$. The registrar's actions are modelled by process $ProcR$, as follows.

$$\begin{aligned} ProcR = & \sum_{v \in \mathcal{V}} r(v, R, (\{v\}_{sk(v)})) \cdot s(R, v, envelope_v) \cdot r(v, R, (\text{vot}, \text{onion})) \cdot \\ & s(R, v, \{\text{vot}, \text{onion}\}_{sk(R)}) \cdot s(R, C, \{\text{vot}, \text{onion}\}_{sk(R)}) \cdot ProcR \end{aligned}$$

Counter process

Note that the counter has no interactions with the voter (cf. Figure 5.5). Furthermore, we note that all the actions of the counter are duplicated in the interaction between voter and registrar, except for the sending of those envelopes the voter does not choose. As such, we omit the counter in this limited modelling of PaV.

Limitations of the PaV model

First of all, the above model is focused on voter interaction. It is not a complete modelling of PaV, but suffices to show how the salient details (such as paper ballots, and envelopes) of voter interaction in PaV could be modelled in the framework.

In addition, the model does not capture the provision that the voter destroys the left column of the ballot. In terms of the framework, this would equate to there being no evidence that π_v was used, that is, no link with the envelope. However, a conspiring voter would not destroy the link, unless the system enforced this. Enforcement is done via procedural means in PaV. Therefore, we expect to find that the cryptographic measures of PaV by themselves are insufficient to guarantee privacy in the analysis below, and thus we expect the analysis will find a privacy risk.

5.2.3 Voter control of privacy in PaV

We claim that the above model of the technical aspects of PaV offers privacy, but does not enforce privacy. Thus, auxiliary procedures and physical requirements are necessary to ensure resistance to coercion.

To prove privacy of PaV, we prove that in any non-trivial setting, the choice group of any non-trivial choice function γ_1 contains more than one element. This implies that PaV does offer at least some privacy in non-trivial settings.

Lemma 4 (privacy of PaV). *Suppose $|\mathcal{V}| > 1$ and $|\mathcal{C}| > 1$. Then for any choice function γ_1 such that there are voters $va, vb \in \mathcal{V}$ for whom $\gamma_1(va) \neq \gamma_1(vb)$, we have $|cg(PaV, \gamma_1)| > 1$.*

Proof. The proof unfolds similar to the proof of Lemma 2. Thus, we take a choice function γ_1 and a trace $t_a \in Tr(PaV^{\gamma_1})$ and show that there exists a choice function $\gamma_2 \neq \gamma_1$ such that there exists a trace $t_b \in Tr(PaV^{\gamma_2})$ for which $t_a \sim t_b$.

We pick $va, vb \in \mathcal{V}$, $ca, cb \in \mathcal{C}$, and γ_1, γ_2 such that $ca \neq cb$ and $\gamma_1(va) = \gamma_2(vb) = ca$ and $\gamma_1(vb) = \gamma_2(va) = cb$. We examine traces $t_a \in Tr(PaV^{\gamma_1})$ and $t_b \in Tr(PaV^{\gamma_2})$, where every agent (including va, vb) executes in the same order. As a receive event reveals no more information than the corresponding send event, we need only consider one of the two. We choose to focus on the send/receive events of voter va . For the send/receive events where the voter is not involved, we focus on the counter.

In trace t_a , we first focus on the send/receive events of voter va . Suppose va was instantiated as $ProcV(va, sk(va))$. The intruder has the following view of the events in t_a and the events that occur at the same index in t_b (labelled for convenience):

	$obstr(t_a)$	$obstr(t_b)$
	\vdots	\vdots
1.	$s(va, R, \{va\}_{sk(va)})$	$s(va, R, \{va\}_{sk(va)})$
	\vdots	\vdots
2.	$r(R, va, \{\pi_{va}, \{\pi_{va}\}_{pk(C)}\}_{pk(va)})$	$r(R, va, \{\pi_{va}, \{\pi_{va}\}_{pk(C)}\}_{pk(va)})$
	\vdots	\vdots
3.	$s(va, R, (\pi_{va}(ca), \{\pi_{va}\}_{pk(C)}))$	$s(va, R, (\pi_{va}(cb), \{\pi_{va}\}_{pk(C)}))$
	\vdots	\vdots
4.	$r(R, va, \{\pi_{va}(ca), \{\pi_{va}\}_{pk(C)}\}_{sk(R)})$	$r(R, va, \{\pi_{va}(cb), \{\pi_{va}\}_{pk(C)}\}_{sk(R)})$
	\vdots	\vdots

The events of voter $vb = \text{Proc } V(vb, sk(vb))$ are equivalent for the obvious substitution. Notice that events 1 and 2 are equivalent in t_a and t_b . Furthermore, notice that the difference between event 3 in t_a and in t_b is $\pi_{va}(ca)$ vs. $\pi_{va}(cb)$. The same holds for event 4. Thus a reinterpretation that interprets $\pi_{va}(ca)$ as $\pi_{va}(cb)$ suffices to reinterpret t_a as t_b with respect to the events of voter va . The similar case is made for voter vb . Below we describe the reinterpretation ρ_a such that $\rho_a(t_a) = t_b$.

$$\begin{aligned} \rho_a(\pi_{va}(ca)) &= \pi_{va}(cb) && \text{voter } va, \text{ event 3} \\ \rho_a(\pi_{vb}(ca)) &= \pi_{vb}(cb) && \text{voter } vb, \text{ event 3} \end{aligned}$$

Thus, we have $\rho_a(t_a) = t_b$. As we made no restrictions on t_a , we derive $t_a \sim t_b$ from this in the same fashion as in the proof for Lemma 2. \square

Lemma 5 (voter-controlled privacy of this model of PaV). *For any non-trivial setting (i.e. $|\mathcal{V}| > 1, |\mathcal{C}| > 1$), the following holds:*

1. *PaV is not conspiracy resistant for conspiring behaviour of type 1.*
2. *PaV is not conspiracy resistant for conspiring behaviour of type 2.*

Proof. The proof follows the lines of the proof of Lemma 3. Here, we only highlight the important differences with that proof.

The rewriting operation for type 2 conspiracy leads to the following voter process.

$$\begin{aligned} \Theta_1(v, \text{Proc } V(v, sk(v))) &= \mathbf{is}(v, \mathbf{knw}_v) \cdot s(v, R, (\{v\}_{sk(v)})) \cdot \\ &\quad r(R, v, \{\pi_v, \mathbf{onion}\}_{pk(v)}) \cdot \\ &\quad s(v, R, (\pi_v(\mathbf{vc}), \mathbf{onion})) \cdot \\ &\quad r(R, v, \{\pi_v(\mathbf{vc}), \mathbf{onion}\}_{sk(R)}) \cdot \delta \end{aligned}$$

In type 2 conspiracy, $sk(v) \in \mathbf{knw}_v$. Therefore, the intruder can open message 2 $\{\pi_v, \{\pi_v\}_{pk(C)}\}_{pk(v)}$, and thus the intruder knows which permutation π_v was used. Given knowledge of π_v , events 3 and 4 cannot be reinterpreted.

The same reasoning holds for type 1 conspiracy (which shares final knowledge). Furthermore, in type 1 conspiracy, $\pi_v \in \mathbf{knw}_v$. Thus, the voter can share the candidate order directly with the intruder. Note, however, that a voter-supplied candidate order π_v can be reinterpreted unless there is proof that the voter received this order. In this case, event 2 can supply this proof, but only if the intruder can “open the envelope”, that is, decrypt the message, or if the intruder can recreate the envelope using π_v . \square

5.2.4 Discussion

While the proof of Lemma 4 shows that PaV can offer a degree of privacy, the proof of Lemma 5 indicates that the permutation π_v , in this model of PaV, is a source of privacy concerns. There are two possible reasons for this: the model is incorrect, or the system has a weakness. In this case, both reasons are applicable. Modelling envelopes as public-key encryptions ties a permutation π_v to a specific voter via message 2. This link also present in the actual PaV. However, there the risk of exposing this link is mitigated by procedural means (shredding π_v).

This has an impact on adapting PaV for remote voting, such as via the Internet. While cryptography can be easily used in a remote voting setting, a remote voting setting lacks a controlled environment and thus cannot enforce such a procedure. This means that using PaV in a remote voting setting like this carries a privacy risk: the candidate order can be linked to a voter. To avoid this, communication of the candidate order to the voter should be reexamined. One approach is to prove the candidate order to the voter using designated verifier proofs. To prevent any information leakage, such proofs should be communicated over untappable channels. In addition, to ensure that *only* the voter learns the candidate order, it is imperative that the candidate order is shuffled by multiple, independent parties. A voting system along these lines was proposed before PaV's inception by Hirt and Sako [HS00]. The fact that our sketched approach converges on their work supports the feasibility of this approach.

The analysis of PaV illustrates that the framework can be used to determine privacy voting systems that incorporate physical as well as procedural measures. Furthermore, the analysis indicated the privacy risks of PaV, which hamper its use for remote voting. Combining this insight with the knowledge of privacy primitives as discussed in Section 3.1 enabled us to sketch a possible direction for adapting PaV to a remote voting setting. In the next section, we illustrate how the concept of choice groups enables quantified reasoning about privacy. This is done by means of the 3BS voting scheme, which provides an interesting trade off between privacy and verifiability.

5.3 Privacy and verifiability

In Chapter 3, we saw that various voting schemes approached reconciling privacy and verifiability in a flawed manner. Verifiability seems contrary to privacy: verifiability requires some matching between voters and votes, which seems detrimental to privacy. This section offers a high-level reasoning, using the concept of choice groups, of how these two security principles interact.

Chevallier-Mames et al. [CFS⁺06] show that universal verifiability is incompatible with privacy properties, specifically, with unconditional privacy and receipt-freeness. They show that in their setting,

- verifiability and unconditional privacy cannot be achieved simultaneously (in their setting) unless all voters vote, and
- verifiability and receipt-freeness cannot be simultaneously achieved.

Their basic setting does not incorporate phases and lacks assumptions on communication (such as untappable channels). This substantiates the conjecture of Hirt and Sako [HS00] that physical assumptions are necessary to ensure receipt-freeness.

Chevallier-Mames et al. define unconditional privacy as follows: the distribution of votes is independent of any publicly available information except for the result of the election. The authors show that in their setting, this is not compatible with universal verifiability.

However, there is room to reconcile a form of privacy with verifiability – even without physical assumptions. The framework presented in this chapter quantifies privacy.

One of the advantages of the framework is that privacy-reducing attacks can now also be distinguished. Thus, while unconditional privacy may not be achievable, we conjecture that it is possible to satisfy universal verifiability while retaining some privacy, more specifically, while still satisfying $|cg(\mathcal{VS})| > 1$.

To support this, we examine the 3BS system [RS07], which reduces, but not nullifies, privacy and provides a certain level of assurance, but not full verifiability.

5.3.1 Privacy and verifiability in 3BS

We briefly recapitulate the 3BS system. In 3BS, a vote is split over three single ballots (see Figure 5.6), which together form one Threeballot. Each ballot carries a unique identifier. To vote for a candidate, a voter ticks two boxes in the row of that candidate; every other candidate-row only receives one tick mark. The voter is free to place the ticks in any column, as long as there is one row with two ticked boxes (her choice) and all other rows have one ticked box.

ballot 1a	ballot 1b	ballot 1c
North ○	North ●	North ○
South ●	South ○	South ○
East ●	East ○	East ●
West ○	West ○	West ●
\$xY63#bZ['@a0~U.3G<]Lw%4!r;}

Figure 5.6: A *Threeballot* in favour of “East”

Upon casting the vote, the Threeballot is validated to be correctly formatted (i.e. one line with two marks, all other lines have one mark). If the Threeballot is formatted correctly, it is separated into three ballots. The voter selects one of the ballots, which the system copies and certifies. This is a full copy, meaning that it includes the voter’s marks as well as the ballot identifier. The voter takes home this certified copy (a receipt), and all three ballots are added to the set of received ballots. No one, can register which ballot the voter copies (the system does not retain this either). After elections, the set of all cast ballots is made public. The result is determined as follows: the number of marks per candidate are summed up, and the number of voters is subtracted (as every voter is obliged to mark every candidate once by the format).

Verifiability

The receipt, which carries the unique identifier, allows the voter to verify that one of her three individual ballots is in the set of cast ballots. The intruder could try to modify or delete (attack) a ballot from the set. However, no one knows which of the three ballots was chosen to copy as a receipt. So, if the intruder attacks a ballot, his attack is detectable if the attacked ballot happens to have been copied. The attack is not detectable, if there is no receipt for the ballot. Verifiability is thus no longer absolute.

More precisely, consider an election with n voters casting a Threeballot. Then, we have a set with $3n$ ballots. Of these, $2n$ are not receipts. If the intruder attacks one ballot, the chance of that ballot not having been copied as a receipt is $\frac{2n}{3n}$. After this, there are $3n - 1$ ballots of interest to the intruder left. $2n - 1$ of these are not receipts. Hence, the chance of the intruder attacking a second ballot which was not a receipt is $\frac{2n-1}{3n-1}$. The chance that both ballots are not receipts is thus $\frac{2n}{3n} \cdot \frac{2n-1}{3n-1}$. In general, if the intruder removes k ballots, the chance of him not choosing a receipt is

$$\prod_{i=0}^{k-1} \frac{2n-i}{3n-i} = \frac{\binom{2n}{k}}{\binom{3n}{k}}.$$

Note that this is bounded by $(\frac{2}{3})^k$ from above (the largest term in the expansion of the product) and by $(\frac{2n-k+1}{3n-k+1})^k$ from below (the smallest term in the expansion of the product). Therefore, the chance of the intruder affecting k ballots without affecting a receipt is at most $(\frac{2}{3})^k$, which is less than 1% for $k \geq 12$. While 3BS does not offer absolute verifiability, it does offer a reasonable form of verifiability.

Privacy

Next, we turn to privacy of the 3BS system. Given the specific way of voting in 3BS, only a limited subset of the cast ballots can form a valid Threeballot with a given receipt (to be more precise, only those ballots combined with the receipt such that there is only one row with two tick marks). For example, consider a receipt with a tick mark for every candidate. This can only be matched with one entirely blank ballot, and one ballot containing precisely one tick mark. Moreover, the entire set of received ballots consists only of valid Threeballots. As a result, some valid combinations can be ruled out as Threeballots. This is because these combinations leave an inconsistent set of ballots: a set in which at least one ballot does not form a valid combination with any two of the other ballots. Thus, some valid combinations must consist of ballots from more than one Threeballot.

Conspiring voters. An obvious attack on vote privacy, already pointed out by the designer, is to agree a priori with the intruder on how to fill in the ballots (captured by class b conspiracy). The intruder can then easily verify if all three ballots are cast. However, the voter may collude with other voter(s) to ensure that together, they cast the required ballots.

For example, suppose the intruder instructs voter va to produce one blank ballot, one ballot marking every candidate and one ballot marking only $c3$. This seems a vote for $c3$. However, with the aid of voter vb , voter va can satisfy the intruder requirement and remain free to vote $c2$, while vb votes $c1$, as shown in Figure 5.7. Note that $c3$ received 0 votes in Figure 5.7. Thus, even with this level of conspiracy, the choice group of the voter may contain more than one element, i.e. $|cg_v(\Theta_b(v, 3BS), \gamma)|$ is not necessarily 1.

Another possibility is for the voter to reveal her receipt after the elections (class 1 conspiracy). As mentioned above, not every combination of three ballots constitutes a valid Threeballot. For example, in Figure 5.7, some of the possible combinations

ballot va	ballot va'	ballot va''	ballot vb	ballot vb'	ballot vb''
c1 ●	c1 ○	c1 ○	c1 ●	c1 ●	c1 ○
c2 ●	c2 ○	c2 ●	c2 ●	c2 ○	c2 ○
c3 ○	c3 ●	c3 ○	c3 ●	c3 ○	c3 ○
id va	id va'	id va''	id vb	id vb'	id vb''

Figure 5.7: Faking conspiracy in 3BS

result in valid Threeballots, other combinations are invalid. In Table 5.1, we show a few combinations, their validity, the preferred candidate of the combination and whether the remaining ballots constitute a valid Threeballot as well.

combination	valid	vote for	remainder valid
$va\ va'\ vb'$	yes	c1	yes
$va\ va'\ va''$	yes	c2	yes
$vb\ va'\ vb''$	yes	c3	no
$va\ va'\ vb$	no		
$va\ va'\ vb''$	no		

Table 5.1: Validity of various possible Threeballots

Note that the two ways shown in Table 5.1 to construct two valid Threeballots are the only possibilities to assign every ballot in the set to valid Threeballots. All other combinations leave the set of remaining ballots in an inconsistent state. As such, the intruder does not need the results to infer that no voter voted c3.

If voter va shares her receipt, the intruder can classify the set of ballots according to which pairs of ballots together with the receipt constitute valid Threeballots. This rules out all valid Threeballots that do not contain the receipt. Thus, this reduces the set of ballots attributable to the voter, and so reduces her privacy. In the setting of Figure 5.7, a receipt of ballot va'' determines precisely who voted for which candidate. In this case, the choice group of the conspiring voter $cg_v(\Theta_1(v, 3BS), \gamma)$ is reduced to a singleton.

As pointed out in [RS07], 3BS can also be used in elections where voters are allowed to vote for multiple candidates. In this case, a Threeballot may contain multiple rows with two tick marks. This means that the number of ballots forming a valid Threeballot with a given receipt is increased. As the number of valid combinations directly affects privacy, voter privacy is improved. In the framework, this improvement is precisely captured by the size of choice groups.

3BS's approach to privacy is based on the idea of dividing voter-privacy up into various parts, and only give the voter control over one part. Even though the link between the voter's part of privacy and other parts cannot be severed completely, it can be successfully obscured.

Concluding remarks

3BS shows that it is possible to combine a certain level of privacy with a certain level of verifiability. Thus, by sacrificing some privacy, some verifiability can be achieved. However, the trade-off between these two is not made precise yet. This is where the framework presented in this chapter comes in: for any system that proposes a trade-off between the two, the framework makes the effects on privacy explicit.

5.4 Conclusions

This chapter served to illustrate how to use the framework and the concepts introduced in Chapter 4 to reason about and determine privacy the framework to voting systems and voting schemes. To this end, we determined the privacy of the FOO92 scheme, we examined privacy for the Prêt à Voter system and discussed the trade-off between privacy and verifiability inherent in 3BS.

The FOO92 voting scheme offers privacy, which the framework captures as a non-trivial choice group. Furthermore, FOO92 is not resistant to voter conspiracy. The identification of the weaknesses enabled us to sketch an approach to improving FOO92.

The PaV voting system relies on several procedures and physical measures to ensure privacy. The analysis of PaV clearly points out weaknesses in the technical measures to ensure privacy. The analysis indicates that this privacy weakness is centered on enforcing the secrecy of the candidate order on the ballot. Using this insight, we can think of ways to mitigate this in remote voting settings. We find that our adaptations to PaV for remote voting lead to a system remarkably similar to the one proposed by Hirt and Sako.

Privacy seems to be at odds with verifiability. Using the concept of choice groups, we examined the 3BS system. The 3BS system provides an interesting trade-off between verifiability and privacy. By reasoning about privacy in a quantified manner, it is evident that while privacy is not nullified, it is reduced in 3BS.

These analyses show how the concepts and the framework of Chapter 4 improve the understanding of privacy in voting, and how this improved understanding may be used to improve voting systems.

Part II

Fairness in Digital Exchange

Introduction to DRM

This part of the thesis investigates the domain of Digital Rights Management (DRM), and the security problems relevant to it. The analysis in this chapter results first defines the concept of a DRM system and its core stakeholders. The incentives of the core stakeholders give rise to high-level security properties. The interaction between the core stakeholders is captured by an iteratively refined process model. The combination of the resulting generic process model with the established security properties for the core stakeholders leads to a set of core security requirements for DRM systems.

We conclude by noting that DRM systems have focused on a client-server setting, but that the concept of DRM does not require this. As such, requirements which can be addressed by assumptions in a client-server setting, need to be ensured in other settings.

6.1 Introduction

There is a precarious balance between dissemination of information (to the general public) and stimulation of innovation and art. The easier it is to spread new information, the less possibilities to profit there will be for innovators to reap the fruits of their labor. On the other hand, spreading innovation and art is considered beneficial to society.

The introduction of computers has had a profound impact on this balance. With computers, it is trivial to create a perfect copy of *content* – a term used to indicate a work of art, such as music, literature, movies, etc. This, coupled with the widespread availability of broadband Internet connections, means that completely new venues for spreading content to the public at large have come into existence. This enables a business model, that consists of selling and delivering digital versions of content online. The main point of concern for such a business is to prevent unsanctioned redistribution of the delivered content.

DRM systems address this goal. The purpose of a DRM system is to enable trade in digital content by governing access to the content. Access control is regulated by licenses, and content is only accessible under the terms of a license for that content.

In recent years, there has been a strong push into the research and development of areas encompassed by DRM, such as secure storage [SV02], traitor-tracing [KY03, SW02], watermarking [CBM01], fingerprinting [HK02, PT01] and tamper resistant code [HMST02, CA02]. There have also been various proposals for models of DRM

systems such as a wide, encompassing functional model [Gut03], and various proposals of systems with specific properties, such as domain-based DRM [PKCT04], and interoperability [OMA04, SNB⁺03]. In fact, a quick look at the three most recent ACM DRM conferences [YKS06, YKS07, HJ08] reveals a variety of subtopics in DRM research, ranging from watermarking to legal aspects, from software protection to biometric authentication, from rights expression languages to formal models.

This diversity in the field has led to a situation where there is no uniform view of the *core* security requirements for a DRM system. Even works proposing systems tends to focus upon enabling specific properties (such as interoperability). Often, requirements ensuring core DRM functionality receive a lesser treatment or are not even made explicit. Thus, the set of requirements that ensure core DRM functionality is scattered.

There are several reasons to make this core explicit. The first and foremost reason is that security is an enabling factor for DRM systems. DRM systems are designed to provide a solution for a security problem. An understanding of (the justification for) the core security requirements is crucial for fundamental comprehension of the security of DRM systems. Without a list of core requirements, it is unclear if all requirements are addressed properly by a system – it is even unclear if research efforts are covering all core requirements. Moreover, knowledge of the core security requirements is instrumental in the construction and verification of DRM systems. Such knowledge enables developers to better understand the consequences of security trade offs. In practical systems, such trade offs between desired features and security requirements are not uncommon.

This chapter uses a structured approach to identify core security requirements and provide a justification for them. The goal of this chapter is focused on the security aspects of the design of DRM systems and our aim is to systematically derive core security requirements for DRM systems. Although there is a wealth of methodologies supporting system analysis and design, the methodologies for deriving security requirements are only at their infancy. Therefore, our research will start by identifying some useful methodologies and combining their strengths. In order to provide a base for the found security requirements, we describe a model which limits itself to the core processes of a DRM system. The generic nature of this core model of DRM functionality implies that requirements found for it are applicable to most DRM systems. The extensibility of this core model indicates that it can be augmented to accommodate additional functionality, and, therefore, that this model suffices for our needs.

The rest of this document is organised as follows: Section 6.2 details the approach we use to arrive at our security requirements, resulting in a list of security properties and a generic process model of DRM systems. These form the basis of the security requirements in Section 6.3. We summarise the chapter and present the research question for this part of the thesis in Section 6.4.

Note. In recent times, the idea of using DRM to govern access to content is becoming more and more obsolete, and more and more companies are abandoning the use of DRM for this purpose. However, tentative attempts are being made to apply DRM concepts in other areas, such as electronic health records (see e.g. [NZJK06]) and ubiquitous computing (see e.g. [FSS05]). However, in order for such reapplica-

tions of DRM to be successful, it is paramount that the underlying requirements of DRM are understood, as these are to be incorporated in new fields. Thus, while the original purpose for DRM is being abandoned, the technology is *not* – and that makes understanding the requirements more important than before.

6.2 Domain analysis

To establish the core security requirements of DRM systems, we establish the desired security properties by a domain analysis and use these as the foundations for the security requirements of DRM systems. Note that in a DRM system, involved parties do not seek to reach a common goal, but have selfish stances. As a necessary consequence, the analysis examines each involved role carefully.

The domain analysis consists of three steps. The first step establishes core stakeholders and their incentives. The second step is to derive the desired security properties from the incentives. Finally, the third step is to derive a process model. The process model is to capture the core operations occurring in DRM systems.

6.2.1 Establishing stakeholders and their incentives

The first step in deriving the security requirements is a *stakeholder analysis*. The purpose of this step is to determine the individuals (or roles) who have an interest in using a DRM system, and their incentives for participating. An understanding of their incentives is important, as these incentives lead to security requirements.

To establish the stakeholders and their incentives, a method similar to and inspired by several existing methodologies from the field of Information Systems has been used. We base our research on a variety of methodologies, such as domain analysis (see e.g. [Pri90]), stakeholder analysis (for an overview see [Pou99]), system decomposition (see e.g. [Som04]). Normally, these methods assist in designing a system. However, our goal here is not to design a system, but to focus upon a system's security aspects. Parts of these methods are accordingly adapted to capture the security aspects of DRM systems.

The domain analysis identifies the three core tasks of a DRM system to be content creation, distribution and usage. As a result, the core stakeholders of DRM systems are the content creator (e.g. media companies, artists, etc.), the distributor (e.g. stores) and the user. Table 6.1 depicts the core stakeholders.

<i>Stakeholder</i>	Role
<i>Content creator</i>	Creation of content
<i>User</i>	Acquisition of content
<i>Distributor</i>	Intermediary between users and content creators

Table 6.1: Core stakeholders for DRM systems

Each of these stakeholders has their own, specific incentives for using DRM. As noted before, DRM systems enable trade of digital content by providing users with re-

stricted, license-controlled access to content. This observation together with a generalisation of the stakeholders' roles leads to the following formulation of the core functionality of a DRM system.

Concept 2 (DRM system). *A DRM system is a system that allows users to access digital content according to access conditions as specified in a license. Licenses and content are distributed by distributors, who in turn acquire content from content creators.*

The above description distinguishes three types of core participating roles: the content creator, who creates content; the distributor, who licenses content; and the user¹, who desires access to content. Some other roles, such as network operator, are necessary for an operational DRM system. However, their impact on the functioning of the system does not touch upon the core of the system. Thus, we do not study their security requirements in this chapter. As such, network security and e-commerce security are out of the scope of the current description.

The analysis indicates that the roles of content creator and distributor are executed by companies (e.g. media companies). As companies are legal entities, they are firmly embedded in a legal framework and thus there exist numerous non-technical solutions to ensure that companies adhere to access agreements. This observation holds less for individual persons, which allows for an asymmetry in DRM systems: there are fewer deterrents to prevent individuals from circumventing agreements than there are deterrents preventing companies. Hence for individuals strict technological enforcement of access agreements between them and companies is necessary. This leads to a natural tendency in DRM development to focus on enforcing access control at the user's side, and to focus less (or not at all) on access control at the distributor's side.

Each of the core roles has various reasons for taking part in a DRM system, and thus related security concerns. Below, the incentives are outlined per role.

Content creator incentives

The role of content creator is executed by stakeholders that create content, such as media companies. They do not seek interaction with users directly, but are satisfied with leaving this to an intermediary – i.e. a distributor. DRM can be used to enforce the conditions set on content by the content creator for the distributor. However, this can also be dealt with by non-digital means (contracts, agreements between companies). Content creators use DRM technology to support new business models. For instance, they can create a bundle of desired content and other content. Such a bundle could increase the value of the content for users (for example, by including the 'making of' footage), or it could increase revenue for the content creator (for example, by including commercials). Additionally, using DRM technology it is possible to offer a revenue-generating alternative for traditional downloading. This means that DRM technology can open a new market. Finally, offering content online in a digital version means that the per-content overhead costs are low - there is no need for plastic casing, a colourful cover, etc.

¹We prefer the computer science term 'user' over terms such as 'customer' and 'consumer'.

Distributor incentives

The distributor binds content to access conditions specified in a license. This role saves content creators the overhead of negotiating directly with their customers. The distributor can use DRM technology to offer tailor-made access to content to users. On top of that, overhead compared to selling physical media is substantially reduced, because digital content takes up little physical space and the presentation of the content can also be done digitally. By offering a clearly legitimate and known-quality alternative for downloading, the distributor can open up a new market. And lastly, as content is bound to a license created by the distributor, access to content distributed in this way will comply with the access conditions set by the distributor.

User incentives

Users will be drawn to DRM systems because DRM systems offer a legitimate, known-quality alternative to more dubious sources of content. Ease of use is an important consideration in this regard: if use of the system or acquisition of content becomes bothersome, a user might turn back to other sources.

Another advantage that DRM systems can offer users is the possibility to restrict the access to (and thus the cost of) content to precisely what the user wishes.

For users, it is important that DRM offers an improvement upon aspects of existing content distribution channels, otherwise there is no incentive for users to switch to a DRM distribution channel. This can, for example, be in terms of ease of use or availability. Such improvements can be offset by deterioration of other aspects – privacy concerns could turn users away from a DRM system.

6.2.2 Security properties per core role

The second step of the problem analysis consists of deriving the desired core security *properties* of each core stakeholder, by transforming the incentives into security properties of the system. The high-level properties of this section are then made precise as concrete security *requirements* in Section 6.3.

For each core role in DRM systems a translation of the stated incentives into desired properties is provided. There are various security solutions that come into play in this process (such as a payment infrastructure). We confine our examination to the core of a DRM system, i.e. those required by the core roles for basic functionality of a DRM system, as described in the above concept.

As the intent of this section is to establish which security properties are desired for DRM systems, properties outside the scope of DRM systems (such as those governing negotiations between parties, those governing the privacy of the business operations of distributors, etc.) are not considered below. Properties of DRM systems that are not security related (e.g. functional properties) are also not examined.

Security properties for the content creator

Of paramount importance for content creators is that they receive compensation for access to content they create. Since digitising content facilitates widespread copying of content, content creators need to be assured that the DRM system that safeguards their content will only allow access in exchange for compensation.

Ensuring that the distributor complies with the terms set by the content creator is arranged outside of the DRM system (e.g. by contracts). As DRM systems focus on enforcing access control at the user's side, the specific security requirements of content creators are delegated to distributors. Hence, content creators require simply that access occurs only via the system. More precisely:

- (c1.) Content is only accessible by untampered components created by the official system developers, under the conditions of a valid license issued by a bona fide distributor.

Security properties for the distributor

The distributor acts as an intermediary between users and content creators. The content creator has delegated responsibilities for content protection to the distributor. The role of the distributor is to distribute content and rights to users. The conditions under which a user has access to content are specified in a license. Naturally, a distributor expects the DRM system to enforce this license. Furthermore, it is in the distributor's interest to be able to deliver what users want, when they want it – otherwise they may opt for an alternative distributor. Finally, attacks disrupting or twisting communications with users must be prevented. The above is formulated more precisely as follows.

- (d1.) Content is only accessible by a renderer with a valid license issued to that renderer, originating from the distributor, under the terms stated in that license.
- (d2.) The DRM system precisely delivers the content that has been requested, with the license as requested, in the correct format, at the desired time to the user.
- (d3.) No other party can disrupt or alter the distributor's side of communications with the user.

Security properties for the user

Users have the option of using traditional distribution channels and using a DRM system. In order for a DRM system to be appealing to users, it is important that a user has the impression that the benefits of using DRM outweigh any more negative aspects compared to traditional distribution (e.g. a purchase in a store). Users will thus expect to be able to download content anywhere, at any time. When buying content through traditional channels, users can remain anonymous (up to a certain

degree) – in a digital setting it is much easier to link data to users, violating their privacy. Finally, in traditional distribution, a user is in control of her side of the trade communications (which content, accessible under which conditions, for which price, when does she hand over money, ...) when negotiating a purchase. This should also hold in a DRM setting. Expressing these points exactly leads to the following list.

- (u1.) The user can precisely acquire a consumable form of the content that the user desires, at the moment the user desires it.
- (u2.) Neither content nor licenses can be linked to the user.
- (u3.) No other party can disrupt or alter the user's side of communications with the distributor.

Note that property *u1* coincides with property *d2*. From here on, these two properties are treated as one property.

The complete list of all security properties is depicted in Table 6.2 below. Together, these properties form a solid foundation for the core functionality of DRM systems – and therefore, they also provide sound underpinnings for the security properties of DRM systems. These properties are necessary for DRM systems. Sufficiency depends on the completeness of the descriptions in Section 6.2.1, and on correctness of the translation of these incentives into the expressed security properties.

<p>content creator</p> <p>c.1 Content is only accessible by untampered components created by the official system developers, under the conditions of a valid license issued by a bona fide distributor.</p> <p>distributor</p> <p>d1. Content is only accessible by a renderer with a valid license issued to that renderer, originating from the distributor, under the terms stated in that license.</p> <p>d2. Precisely deliver what has been requested, in a consumable form, at the desired time for the licensee.</p> <p>d3. No other party can disrupt or alter the distributor's side of communications with the user.</p> <p>user</p> <p>u1. Precisely acquire a consumable form of the content that the user desires, at the moment the user desires it.</p> <p>u2. Neither content nor licenses can be linked to the user.</p> <p>u3. No other party can disrupt or alter the user's side of communications with the distributor.</p>
--

Table 6.2: Security properties as desired per role

We take the view that all security properties stated in Table 6.2 are essential to satisfy the incentives of the core stakeholders. However, not all DRM systems will necessarily

seek to address all of the security properties desired by users. For example, there are no compelling reasons for distributors to ensure user privacy (requirement u2). On the contrary, usage data is valuable for creating profiles (which can then be sold). We believe, however, that privacy is an important security, especially in e-commerce systems. Therefore, we feel that the inclusion of privacy as a core incentive is justified.

6.2.3 Conceptual process model

The second step of the problem analysis consists of the development of a *conceptual process model*. This model relates the basic processes in the DRM system to each other, and can be refined to provide a basis for identifying security requirements. This process model is then combined in Section 6.3 with the list of security properties to establish core security requirements of DRM systems.

To ensure that the model is applicable to as many DRM systems as possible, it must be as generic as possible. In order to derive such a generic model, we start with one component for the three core roles in a DRM system: the content creator, the distributor and the user, see Figure 6.1. Components for non-core roles are left out as they can be introduced when necessary, as such additions constitute refinements of the core model. The resulting model is subsequently refined to incorporate various specifics of DRM systems.

The model of Figure 6.1 is read as follows: the creation process creates content, which is forwarded to the distributor. The distribute process distributes the content to the user. The rendering process at the user side renders the content.

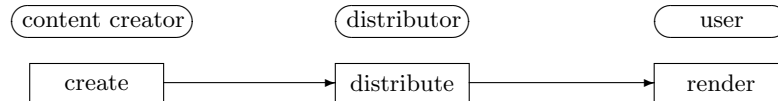


Figure 6.1: Basic process model

The first refinement to this model is instigated by the observation that for content to be enjoyed by users, it is eventually transformed into an analogue form. This means that there exist two variants of content: digital content (which the system is to protect), and analogue content (which a user can consume). As all DRM systems deal with both variants, the generic model can be refined to incorporate this distinction without harming its generality. This results in splitting the rendering process into two processes: one process to convert digital content to analogue content (*content rendering*), and one process to extract the digital content from the DRM bundle (*extract*). This refinement is depicted in Figure 6.2. Note that in general, rendition is the final step in the process. Hence, content rendering is depicted as the last process in the model.

The next refinement is prompted by noting that DRM-protected content cannot be accessed without a license. This does not imply that a license must be bundled with the content in all cases, or even that a license must exist when the content

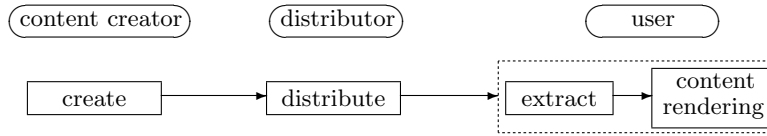


Figure 6.2: Model incorporating rendering of content

is protected. It suffices that the resulting content is not accessible without a valid license. To express the possibility of licenses and protected content existing separately, the distribution process is further refined into two parts. The *content protection* part provides the user with protected content, while the *license creation* part provides the user with a license for protected content. This leads us to the conceptual model depicted in Figure 6.3.

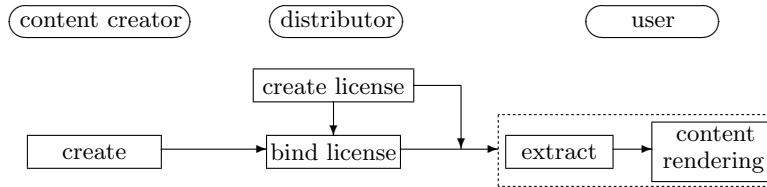


Figure 6.3: Generic process model of DRM

This generic process model now incorporates the core roles and models the core processes in a DRM system. The model can be refined to comply with virtually all existing DRM architectures. For example, additional roles like content issuer and license issuer can be incorporated to comply with the functional architecture of the Open Mobile Alliance DRM architecture [OMA04].

The goal of this chapter is to establish core security requirements of DRM systems. As further refinements constrain the applicability of the model, further refining the model would detract from the core of DRM systems. As such, the model is not further refined.

6.2.4 Results of the problem analysis

The problem analysis has provided two results: a list of security properties desired by the core stakeholders, and a conceptual process model of the processes taking place in a DRM system. Taken together, these results elucidate security requirements of the core functionality of a DRM system.

Completeness of these descriptions has a large impact on which security requirements are found. After all, the more complete the incentives and the core processes are described, the more complete the derived security requirements will be. The methods

we adapted to arrive at the given descriptions support a systematic derivation of security requirements, but do not guarantee completeness of the results. Despite this, we believe that, due to the systematic approach, the descriptions are sufficiently exhaustive for our goals.

6.3 Core security requirements

In this section, the security properties are used as a basis for establishing the security requirements per stakeholder upon the generic process model. Both the process model and the incentives were deliberately kept widely applicable, so that the derived security properties and requirements are generically applicable to the setting of DRM systems. In this way, further detailing of the model will instigate further detailed security requirements.

To refine the security properties, we applied a technique based on Schneier's attack trees [Sch00]. We adapted this technique to result in *objective trees*, which enable systematic analysis of defensive aspects of a system. The results of this systematic analysis are presented below.

6.3.1 Security requirements for the content creator

The desired security properties for the content creator have been established as follows in the previous section:

- (*c1.*) *Content is only accessible by untampered components created by the official system developers, under the conditions of a valid license issued by a bona fide distributor.*

In Figure 6.3, we see that the creator's role in the system is limited to communicating content to the distributor. As noted earlier, DRM systems can be asymmetric: the relationship between a content creator and a distributor is not on par with the relation between a distributor and a user. To support this asymmetry in our generic security requirements, we have chosen not to mention security requirements twice (once for the content creator and once for the distributor), but to only formulate them for the distributor. As mentioned in the introduction, to model a symmetric DRM system it suffices to refine the requirements and the process model.

As can be seen from Figure 6.3, the content creator communicates the content to the distributor. That means, that in order to uphold property *c1*, the end-point of this communication must be an authenticated distributor (*c1.1*), that the content must be protected during communication (*c1.2*), and that the system ensures the security of the content from that point on (*c1.3*). Note that this latter requirement coincides with the first property for the distributor.

6.3.2 Security requirements for the distributor

Each of the established security properties for the distributor is analysed below to arrive at security requirements. The first property was stated as:

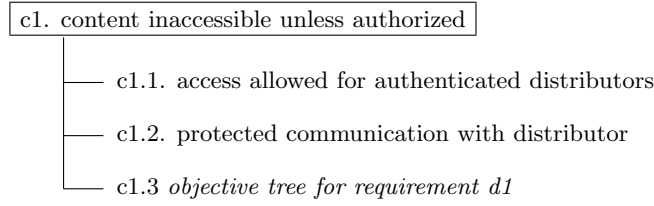


Figure 6.4: Objective tree for property c1

- *(d1.) Content is only accessible by a renderer with a valid license issued to that renderer, originating from the distributor, under the terms stated in that license.*

This property of the distributor is equivalent to property *c1* of the content creator. This property governs the fundamental protection of content in a DRM system. Content protection requires secrecy of content. In terms of the generic process model (Figure 6.3), the provided content must remain inaccessible for anyone and any component except the ones specifically allowed to access the content. Furthermore, successful completion of the user-side process only happens in compliance with a valid license, and only by valid, authenticated components.

In concrete terms, the user side components are secured against attacks (*d1.1*). Furthermore, security breaches in the user process must be constrained in scope (*d1.2*). This means that breaches must not be generically applicable to all installations (prevent Break Once, Run Everywhere, *d1.2a*), and that it must be possible to update system components (*d1.2b*). To ensure that content is not leaked, the processing components all need to be authenticated. Hence, components may only forward content to authenticated components (*d1.3*) and all keys used must remain secret (*d1.5*). Finally, the user side process may only render content if all terms of an appropriate license have been met (*d1.4*). These requirements are depicted succinctly in the objective tree for property *d1* in Figure 6.5. Together, these requirements provide content protection in the generic process model.

Security property *d2* coincides with security property *u1* and is treated with the other user security properties. Hence, we turn to the final security property for the distributor, which was expressed as:

- *(d3.) No other party can disrupt or alter the distributor's side of communications with the user.*

As we can see in Figure 6.3, the distributor's side of communications with users occurs after binding licenses to content. Thus, *d3* is upheld if content is only sent when the distributor desires so (which he does in exchange for receiving compensation, i.e. fair exchange *d3.1*), if the user cannot falsely deny having received content (*d3.2*), and if the communication channel cannot be influenced (*d3.3*). The resulting objective tree is depicted in Figure 6.5.

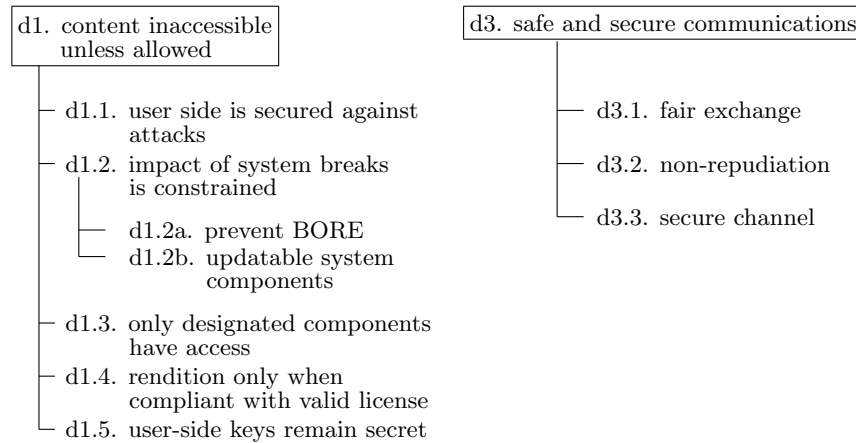


Figure 6.5: Objective tree of properties d1 and d3

6.3.3 Security requirements for the user

Below, the security properties as desired by the user are analysed and security requirements are derived from them. Once again, the generic process model is used as a base to match the stated properties against and objective trees are established to uncover security requirements.

The first security property as desired by the user of a DRM system was established in Section 6.2.2 as:

- *(u1.) Precisely acquire a consumable form of the content that the user desires, at the moment the user desires it.*

This property concerns the delivery capabilities of the system, or, the receiving side of the communication channel between distributor and user in Figure 6.3. In effect, this property states that the communication channel must be available (*u1.1*), integer (i.e. undisruptable and incorruptible, *u1.2*), and that the delivered content and license are those, that were requested (*u1.3*). The latter requirement, and the need to acquire a consumable form of the requested content, cannot be imposed by the channel. They imply that the communication partner (the distributor) must be trusted. Hence it is important that the user is convinced that the party at the other end of the communication channel is indeed the party she trusts, i.e. the distributor. This is achieved by having the distributor authenticate himself to the user (*u1.4*).

The second security property for users of DRM systems is:

- *(u2.) Neither content nor licenses can be linked to the user.*

This property ensures the privacy of a user using a DRM system. The privacy property breaks down according to the generic process model: the user's privacy must be ensured by all parts interacting with the user. In the generic process model (Figure 6.3), these are the distributor, the communication channel from the distributor

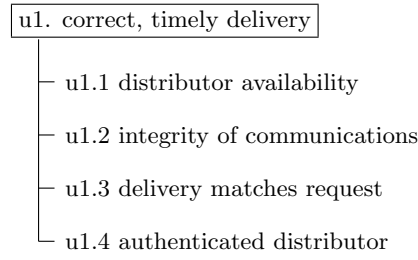


Figure 6.6: Objective tree for property u1

to the user and the user processes of the DRM system. The distributor must handle all personal data he receives with care, and within the limits set by the user (*u2.1*). (Note that EU directive 95/46/EC, the EU directive on “*the protection of individuals with regard to the processing of personal data and on the free movement of such data*” only allows personal data to be kept when necessary for system execution, meaning that this requirement is an EU directive.) Furthermore, the communication channel must not leak private user information (*u2.2*). Finally, the user side of the DRM system must ensure user privacy as well (*u2.3*). The resulting objective tree is depicted in Figure 6.7.

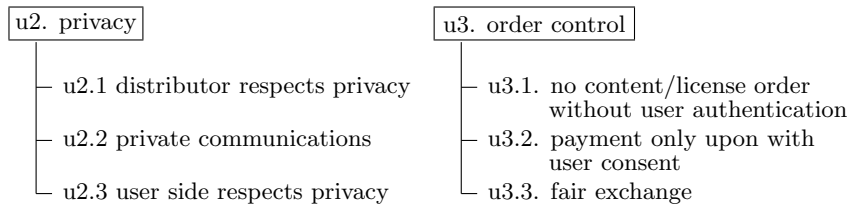


Figure 6.7: Objective tree for properties u2 and u3

The final security property for the user is formulated as:

- (*u3.*) *No other party can disrupt or alter the user’s side of communications with the distributor.*

This property, which is abbreviated as “order control” in Figure 6.7, ensures that the system may only act on the intentions of the user, and not otherwise. In the generic model, this property imposes requirements on the user side of communication between user and distributor. Communication between these parties consists of two distinct parts: the negotiations on compensation for contents and rights on the one hand, and the delivery of content and license on the other. To ensure not other party can act in the user’s stead, the user is required to authenticate herself using her private authentication data for both these parts (*u3.1*). Secondly, to safeguard negotiations, compensation (payment) for content only occurs intentionally (*u3.2*). Finally, the user only pays in exchange for the requested items (*u3.3*). The resulting objective tree is shown in Figure 6.7.

6.3.4 Consequences

The security requirements established in the previous section have some consequences, which are mentioned below.

It follows from the requirements derived from the distributor's desired security properties (d1, d2 and d3) that the components on the user's side should function as a trusted computing base. More specifically, requirements d1.1 and d1.2 require the user-side to be secure, and to remain secure throughout the lifetime of the system. In effect, this requires the system to be able to adapt to attacks.

The points following from property u2.1 indicate that the distributor should at the very least provide a privacy statement. Requirement u1.3 implies that the distributor provides a security policy.

Requirements d3.1 and u3.3 underline the distributor's and user's need to trust that they will receive their rightful due (payment for the distributor, content for the user) in exchange for their goods. If this is not a mutual exchange, the question arises of who receives his due first. How is the second party assured that the first party will deliver if they already received their goods? In effect, users and distributors require an assurance of fair exchange for them to be willing to engage in trade.

Fair exchange in DRM systems is hardly studied. In traditional client-server settings (the focus for most DRM-related research), fair exchange is addressed by the distributor's reputation: the user knows the distributor and trusts in his reputation to provide the content. Consequently, the user is willing to pay before she has received content. As the goal of a DRM system is to enable trade in digitised contents, it seems reasonable that a distributor who uses a DRM system would indeed deliver contents. Nevertheless, the reputation assumption is almost never made explicit in DRM-related literature in client-server settings. As there is little research into DRM systems outside this setting, fair exchange is an open problem for DRM systems.

6.4 Summary

The core security requirements of DRM systems are scattered and unclear. To address this, this chapter established the core security requirements of DRM systems by means of combining a stakeholders' incentives analysis with a generic process model.

Of the resulting list of requirements, requirement *u3.3*, fair exchange, is hardly studied. This is due to the focus on client-server DRM systems, where fair exchange for the client is dealt with by the server's reputation for correct behaviour. However, this leaves unaddressed how to ensure fairness in an environment without a priori known reputations, such as a peer-to-peer setting. In an extreme case, every peer could also act as a redistributor of content, without having an established reputation. Such a setting encompasses the case where a peer is an original distributor and thus seeks to build a reputation. It also encompasses the case where the only incentive for distributing is direct benefit to the distributor.

However, fairness is not the only requirement for DRM systems. DRM systems must safeguard access to content in order to act as successful trade-enabling systems. To this end, they must be rigorously secure. Earlier in this thesis, we already argued that

a rigorous approach to security requires methodology with mathematical precision, such as formal methods. But formal security is not enough, due to requirement *d1.2*: a DRM system must also be practically secure.

Thus, the question is raised whether a formally and practically secure DRM system that enables fair exchange is feasible. To address this question, we execute a feasibility study, charged as follows.

Research question II. *Is a formally and practically secure DRM system that enables content redistribution possible?*

The question is addressed in the next chapter by surveying existing work in DRM in a peer-to-peer setting. From this survey, we base our feasibility study on a basic system that aims to provide redistribution. First, the system and its goals are formalised and verified using model-checking. The flaws discovered by the formal analysis are addressed in a new scheme. Furthermore, we introduce contingency procedures to provide practical security. The new system is verified using model-checking to ensure it satisfies the proposed goals.

Fair exchange in DRM

From the previous chapter, the question of enabling and ensuring secure and fair content redistribution in a peer-to-peer environment emerged. In this chapter, this question is addressed by designing and analysing the security of a peer-to-peer DRM system enabling fair exchange.

7.1 Introduction

As noted in the previous chapter, users require fair exchange in order to be willing to trade. In a peer-to-peer (P2P) setting, users may trade with other users that are unknown to them. Thus, fair exchange should be ensured. Furthermore, the trading must ensure that usage rights are not infringed upon, to ensure that providers are willing to participate. Access control (which ensures adherence to usage rights) has been recognised as an open problem for P2P systems in [DGY03]. The possibilities of P2P networks for content distribution have led to various research efforts, though few (to date) have researched fair exchange in this setting.

In the early proposal of Grimm and Nützel [GN02], content is distributed via a P2P system. Their system does not enforce access control, but uses rewards to entice users to compliance. However, this incentive is based on reselling. A user who does not engage in reselling, thus has no incentive to pay for content.

This is followed by Iwata, Abe, Ueda and Sunaga [IAUS03], who sketch various models for leveraging the advantages of P2P distributions. They distinguish two licensing models for P2P systems: licenses are bundled with the content, and rewritten by users, or licenses are obtained from a central server. They discuss the possibility of resale on an high level, but do not detail a system for this.

Reti and Sarvar propose the *DiMaS* system [RS04b, RS04a]. This system uses P2P distribution as a means to lower the costs and load of distribution over users. License acquisition is done via a central server. Such a stance is seen more often in DRM-related literature, e.g. in the system proposed by Chong and Deng [CD06] and in *Music2Share*, a system proposed by Kalker, Epema, Hartel, Lagendijk and Van Steen [KEH⁺04]. Similar to DiMaS, [CD06] and Music2Share seek to address the bandwidth bottleneck of distributing content by leveraging P2P distribution, while keeping licensing centralised. In contrast, we believe that the idea that users will distribute protected content via P2P exchange for free, and then pay via the DRM system for accessing the content, is optimistic. Furthermore, it also does not leverage the full potential of P2P connectivity.

The potential for resale has been recognised by Adelsbach, Rohe and Sadeghi. Their work focuses only on formalising rights. They formalise rights and rights transfer [ARS05] such that rights can be transferred, although using rights is not part of their model.

Krishnan Nair, Popescu, Gamage, Crispo and Tanenbaum propose a DRM system that allows resale. Their system [NPG⁺05] is based on trusted client devices. In their system, users can also act as content redistributors. This allows users not only to buy rights to use specific content, but also to redistribute content and rights in a controlled manner. Recognising that this is technically challenging to security, the main goal of [NPG⁺05] is to enable content redistribution while resisting systematic distribution rights violation. Similarly, Chu, Su, Prabhu, Gadh, Kurup, Sridhar and Sridhar propose a system that enables reselling of rights [CSP⁺06]. Each right to access (view) content once is represented in this system by a “ticket”. Users receive (some) “credits” upon purchasing tickets, and these credits may be traded for tickets with other users. In contrast, the system proposed by Nair et al. abstracts away from both payment and rights. As such, the link between them (i.e. payment for rights) is to be ensured by the system, while the exact amount of payment for a specific set of rights is left open. Such a more generic approach is more flexible, as both payments and rights can be done to an implementer’s pleasing (e.g. peer-to-peer payments could be done by credits, but also in cash).

The generic nature of the system by Nair et al., hereafter referred to as the NPGCT system, is appropriate for an initial feasibility study. Hence, we base our feasibility study on this approach.

7.1.1 Approach to study feasibility

This chapter approaches the feasibility study in three stages. First, we formally specify the protocols of the NPGCT system. A formal analysis of our specification revealed two security flaws: a rights-replaying flaw and a problem with fair exchange between users. We propose an extended scheme, dubbed *Nuovo DRM*, to address these issues. A formal specification and verification of Nuovo DRM is subsequently presented and (a finite model of) the scheme is shown to indeed achieve its design goals.

Secondly, to study the feasibility of a *formally secure* DRM system, we present the used state-of-the-art formal tools and techniques to handle the problem of verifying the security of DRM schemes. We use the μ CRL process algebraic language [GP95] and tool set [BFG⁺01] to specify the protocol participants and the intruder model. The expressive power and flexibility of the μ CRL language compares favourably to other specification languages. These factors enable us to keep the formalisation close to the actual implementation. Due to the complexity, the size of the scheme and the branching nature of the protocols, generating the state space is a time-consuming process. Several approaches to handle this so-called “state space explosion” exist, such as counter-based abstractions [PXZ02] or parametrised abstraction techniques [PV04]. These techniques are not straightforwardly applicable to our problem however, as they focus on abstracting away state details, which in a DRM setting amounts to abstracting away rights and content – exactly the main points of interest. In order to address state space generation, we resorted to a distributed instantiation of the

μ CRL tool set [BCL⁺07] to generate and minimise the corresponding state spaces. In particular, since the Nuovo DRM scheme is highly non-deterministic due to the presence of several fall-back scenarios, with the inclusion of an intruder model to the system, it easily runs into the limits of single-machine state space generation. To the best of our knowledge, we are the first to formally verify a whole DRM scheme. Moreover, we adapt the standard formal Dolev-Yao intruder model [DY83] to reflect the restricted behaviour of compliant devices in DRM systems (which are not under *full* control of the intruder, even if owned by the intruder).

Finally, to ensure a *practically secure* DRM system, we present a set of mitigation strategies that ensure that the assumptions of the formal verification are reasonably upheld in practice.

7.1.2 Related work in fair exchange

Nuovo DRM introduces an optimistic fair exchange protocol. This class of protocols was introduced in [Aso98] and since then have attracted much attention. The closest fair exchange protocol to our scheme is perhaps the probabilistic synchronous protocol [AGGV05], in that it too relies on trusted computing devices in exchange. In contrast to [AGGV05], the optimistic fair exchange protocol in Nuovo DRM is a deterministic asynchronous protocol that achieves strong (as opposed to probabilistic) fairness, but, as a drawback, it relies on impartial agents to secure unsupervised exchanges.

The formal analysis of Nuovo focuses on analyzing transactional properties of DRM schemes. There are several works in literature on model checking (usually small instances of) optimistic fair exchange protocols, such as [GRV03, KR01, SM02]. What makes our study unique is the size of the system that is automatically analysed as well as the capturing of some DRM-specific features of the system, like compliant devices, in the model. Constraint solving for checking fair exchange protocols proposed in [KK05] can detect type-flaw attacks, but is restricted to checking safety properties. Theorem-proving approaches to checking fairness of protocols [AB03, BP01, ES05] can provide a complete security proof at the cost of heavy human intervention, and thus cannot be easily integrated into the protocol design phase.

7.1.3 Structure of this chapter

We start by explaining the notations and (cryptographic) assumptions used in Section 7.2. Section 7.3 summarises the NPGCT scheme, which provides the basis for our refined scheme. Section 7.4 presents the Nuovo DRM scheme, its assumptions, its goals and its protocols. Nuovo DRM is then formalised in Section 7.5. This model is formally analysed in Section 7.6 and shown to achieve its goals. In Section ?? several mitigation procedures are discussed. Finally, Section 7.8 summarises the chapter.

7.2 Assumptions and notations

Throughout the paper, the following assumptions are used.

Trusted devices assumptions. A compliant device is a tamper-proof hardware device that, though possibly operated by malicious owners, follows only its certified software. We assume that compliant devices are able to locally perform *atomic* actions: multiple actions can be logically linked in these devices, such that either all or none of them are executed. They also contain a limited amount of secure scratch memory and non-volatile storage. These requirements are typically met by current technologies. A legitimate content provider, (albeit abusively) referred to as trusted third party (TTP), is assumed impartial in its behaviour and eventually available to respond to requests from compliant devices.

Cryptographic assumptions and notations. In our analysis cryptographic operations are assumed to be ideal à la Dolev-Yao [DY83]. We assume access to a secure one-way collision-resistant hash function h ; therefore $h(x)$ uniquely describes x . A message m encrypted with symmetric key K is denoted $\{m\}_K$, from which m can only be extracted using K . Notations $pk(X)$ and $sk(X)$ denote the public and private keys of entity X , respectively. In asymmetric encryption we have $\{\{m\}_{sk(X)}\}_{pk(X)} = \{\{m\}_{pk(X)}\}_{sk(X)} = m$. Encrypting with $sk(X)$ denotes signing and for convenience we let m be retrievable from $\{m\}_{sk(X)}$.

Additionally, the following notation is used:

$d1, d2$	compliant devices
P	trusted, legitimate content provider
$owner(d1)$	owner of device $d1$
$c \in Cont$	an content item from the set of all content
$h(c)$	a unique descriptor of item c
$r \in Rgts$	one right from the finite set of all possible rights
$R_{d1}(c)$	the rights stored on device $d1$ for content c

It is assumed that the unique descriptors of all $c \in Cont$ are publicly known.

7.3 The NPGCT DRM scheme

The NPGCT scheme [NPG⁺05] by Nair et al. was proposed as a DRM-preserving digital content redistribution system where a user can act as a content redistributor, without this adversely affecting the protection offered by the DRM scheme. In this section we briefly describe the NPGCT scheme and then present the results of its formal analysis. For a detailed specification of NPGCT see [NPG⁺05].

7.3.1 NPGCT protocols

The NPGCT scheme describes two types of exchanges: a provider-to-client (P2C) exchange, to distribute content from provider P to client $d1$, and a client-to-client (C2C) exchange for client $d1$ to resell content to another client $d2$.

Provider-to-client exchange (P2C)

This exchange is initiated by the owner of $d1$ who wishes to buy item c with rights r from provider P . From [NPG⁺05]:

1. $d1 \rightarrow P$: Request content
2. $d1 \leftrightarrow P$: Mutual authentication, [payment]
3. $P \rightarrow d1$: $\{c\}_K, \{K\}_{pk(d1)}, r, \sigma, \Lambda$
 $\sigma = \text{meta-data of } c, \Lambda = \{h(P, d1, c, \sigma, r)\}_{sk(P)}$

Step 2 in the protocol serves as a placeholder for a multi-stage authentication protocol. Furthermore, in the protocol, Λ acts as a certification that $d1$ has been granted rights r and helps in proving $d1$'s right to redistribute c to other clients. Additionally, Λ binds the meta-data σ to content c , which prevents masquerading attacks on c .

Client-to-client exchange (C2C)

This exchange is initiated by the owner of $d2$ who seeks to buy c with rights r' from $d1$, for which $d1$ holds certificate Λ . From [NPG⁺05]:

1. $d2 \rightarrow d1$: Request content
2. $d1 \leftrightarrow d2$: Mutual authentication
3. $d1 \rightarrow d2$: $\{c\}_{K'}, \{K'\}_{pk(d2)}, R_{d1}(c), r', \sigma, \Lambda, \Lambda'$
 $\Lambda' = \{h(d1, d2, c, \sigma, r')\}_{sk(d1)}$
4. $d2$: Verifies σ, Λ' and $R_{d1}(c)$ using Λ
5. $d2 \rightarrow d1$: $\psi, [\text{payment}]$
 $\psi = \{h(d1, P, \{c\}_{K'}, \sigma, r')\}_{sk(d2)}$

By sending ψ , $d2$ acknowledges that it has received c with rights r' from $d1$, while Λ and Λ' form a certificate chain that helps to prove that $d2$ has been granted rights r' . The acknowledgment ψ was intended to be used in C2C-dispute resolution, although that notion is not further explored in [NPG⁺05].

7.3.2 Results of formally analysing NPGCT

As part of our work, we formally specified and checked the NPGCT scheme. In this section, we present the results of this analysis. The assumptions of the scheme, the security goals it was tested against, their formalisation, the protocol specification tool set and the model checking technology used here are similar to those used for Nuovo DRM, which are discussed in the following sections.

Two security flaws in the NPGCT scheme were revealed by our analysis. First, it was found that in the P2C (and similarly the C2C) protocol, a malicious user can feed rights from a previous session to the trusted device by replaying step 3. This replay is possible because freshness of the authentication phase is not extended to guarantee freshness of step 3 (delivery of the content-right bundle). This flaw allows $d1$ to accumulate rights without paying P for it. As a remedy, fresh nonces from the

authentication phase can be used in Λ to ensure the freshness of the whole exchange, c.f. Section 7.4.

Second, in the C2C protocol, payment is not bound to the request/receive messages exchanged between two devices. Thus, once $d2$ receives c in step 3, the owner of $d2$ can avoid paying $d1$ by quitting the protocol. Since this exchange is unsupervised, the owners of compliant devices are forced to trust each other to complete transactions. While it is reasonable to extend such trust to a legitimate content provider, it should not be assumed for potentially dishonest device owners in C2C exchanges. (Note that fairness in exchange is not a goal of NPGCT.)

7.4 The Nuovo DRM scheme

This section describes the proposed changes to the NPGCT, dubbed Nuovo DRM, which in particular addresses the security concerns identified in Section 7.3.2. This section provides an informal description of Nuovo DRM's goals, assumptions and protocols.

7.4.1 Nuovo's goals

The aim of Nuovo DRM is to enable content redistribution whilst resisting systematic content pirating. Hence, Nuovo DRM provides a secure DRM scheme which encompasses content redistribution. The security of the system is to address normal DRM security concerns as well as security concerns introduced by content redistribution. This is captured by the goals below. We require the Nuovo DRM system to achieve these goals (which are the same as those used to analyse the NPGCT system in Section 7.3.2):

- G1 (effectiveness).** A system provides effectiveness if it terminates successfully when used by honest participants in a secure environment. For a DRM system, this means that a desired content-right bundle is exchanged for the corresponding payment order. Effectiveness is a sanity check for the functionality of the protocol and is therefore checked in a reliable communication system with no attacker.
- G2 (secrecy).** No party may learn any $c \in \text{Cont}$ not intended for him. Usually, content is encrypted for intended receivers. Nuovo DRM (similar to NPGCT) limits the distribution of protected content by encrypting them for intended compliant devices. In this situation, secrecy is attained if DRM-protected content never appears unencrypted to any known non-compliant device. Secrecy covers requirements d1.3, d1.4, d1.5, and u2.2 from Chapter 6.
- G3 (resisting content masquerading).** Content masquerading occurs when content c is passed off as content c' , for $c \neq c'$. Preventing this attack ensures that an intruder cannot feed c' to a device that has requested c . Resistance to content masquerading covers requirement u1.3 from Chapter 6.

G4 (strong fairness). Assume Alice owns an item c_A and Bob owns an item c_B . Informally, strong fairness states that if Alice and Bob run a protocol to exchange their items, in the end either both or neither of them receive the other party's item [PVG03]. Strong fairness usually requires the items exchanged in the system to be *strongly generatable*: in Nuovo DRM, a content provider can provide the exact missing content if the exchange goes amiss. Strong fairness also guarantees *timeliness*, which informally states that, in a finite amount of time, honest protocol participants can safely terminate their role in the protocol with no help from malicious parties. Strong fairness covers requirements d3.1, d3.2 and u3.3 from Chapter 6.

Note that the goals from Chapter 6 not covered by the above goals are covered by the mitigation procedures in Section 7.7. The exceptions are requirements u2.1, u2.3 (privacy) and u3.2 (payment only with user consent), that only apply to parts of a DRM system that Nuovo DRM does not specify.

Properties of systems can be divided into two classes: *safety* properties, stating unwanted situations do not happen, and *liveness* properties, stipulating desired events eventually happen (for a formal definition of these property classes see [Lam77]). Goals G1 (secrecy) and G2 (resistance to content masquerading) are safety properties, while strong fairness is a liveness property. Liveness properties require resilient communication channels (assumption A2 below) to hold (for an in-depth discussion of fairness in exchange, see [Aso98]).

7.4.2 Assumptions of Nuovo DRM

The following assumptions are made regarding the working of Nuovo DRM. Note that assumptions A1 and A2 limit the power of the intruder, as explained further in Section 7.5.2.

A1 (Trusted devices). We assume that devices are compliant. However, owners of compliant devices are untrusted. They may collude to subvert the protocol. They can, in particular, arbitrarily switch off their own devices (the “crash failure model” in distributed computing terminology).

A2 (resilient communication). We assume an asynchronous resilient communication model with no global clock, i.e. the communication media deliver each transmitted message intact in a finite but unknown amount of time. Resilience is necessary when aiming for fairness [FLP85], and is realisable under certain reasonable assumptions [BCT96].

A3 (PKI hierarchy). There exists a hierarchy of public keys, with the public key of the root authority embedded in each compliant device and available to content providers. Using such an infrastructure, a device can prove its identity or verify other devices' identities without having to contact the root. Identities $d1$, $d2$ and P implicitly refer to these authentication certificates issued by the trusted authorities.

A4 (price negotiations). Protocol participants negotiate the price of content in advance. In Nuovo DRM, the price of the content being traded is bundled with the requested rights.

7.4.3 Nuovo DRM protocols

As in NPGCT, our scheme consists of two main protocols: a provider-to-client exchange for provider P and client device $d1$ and a client-to-client exchange for clients $d1$ and $d2$. These protocols derive from the NPGCT schemes, but are updated to incorporate authentication and strong fairness. Strong fairness requires a recovery sub-protocol for the client-to-client exchange.

Provider-to-client exchange (P2C)

The owner of $d1$ wants to buy item c with rights r from content provider P . Here $d1$ and P , but not $owner(d1)$, are assumed trusted. The P2C exchange is shown in Figure 7.1.

-
- | | | |
|----|------------------------------|--|
| 1. | $owner(d1) \rightarrow d1 :$ | $P, h(c), r$ |
| 2. | $d1 \rightarrow P :$ | $d1, n_{d1}$ |
| 3. | $P \rightarrow d1 :$ | $\{n_P, n_{d1}, d1\}_{sk(P)}$ |
| 4. | $d1 \rightarrow P :$ | $\{n_{d1}, n_P, h(c), r, P\}_{sk(d1)}$ |
| 5. | $P \rightarrow d1 :$ | $\{c\}_K, \{K\}_{pk(d1)}, \{r, n_{d1}\}_{sk(P)}$ |
-

Figure 7.1: P2C exchange: $d1$ buying content from provider P

In the first step, the hash of the desired content c (retrieved from a trusted public directory), rights r and the identity of a legitimate provider P are fed to the compliant device $d1$. Following assumption A4, $owner(d1)$ and P have already reached an agreement on the price. Whether P is a legitimate provider can be checked by $d1$ and vice versa (see assumption A3). In the second step, $d1$ generates a fresh nonce n_{d1} and sends it to P , which will continue the protocol with step 3 only if $d1$ is a compliant device. The message of the fourth step completes the mutual authentication between $d1$ and P . This message also constitutes a *payment order* from $d1$ to P . After receiving this message, P checks if r is the same as previously agreed upon (assumption A4) and only if so, stores the payment order (for future/immediate encashing) and performs step 5 after generating a random fresh key K . When $d1$ receives the message of step 5, it decrypts $\{K\}_{pk(d1)}$, extracts c and checks if it matches $h(c)$ from the first message and if n_{d1} is the same as the nonce from the second message. If these tests pass, $d1$ updates $R_{d1}(c)$ with r , i.e. r is added to $R_{d1}(c)$. Note that $R_{d1}(c)$ is not necessarily r : $d1$ could already have some rights associated with c , for instance, acquired from an earlier purchase. Since we abstract away from rights semantics (as discussed in Section 7.1.2), updating rights is left unspecified here.

Client-to-client exchange (C2C)

The owner of $d2$ wants to buy item c with rights r' from another compliant device $d1$. This exchange can be seen as a fair exchange protocol where $d1$ and $d2$ want to exchange a content-right bundle for its associated payment such that either both or neither of them receive their desired items. In deterministic protocols, however, achieving fairness has been proven to be impossible without a TTP [EY80]. Assuming that most participants are honest and protocols go wrong only infrequently, it is reasonable to use protocols which require TTP's intervention only when a conflict has to be resolved. These are called *optimistic* fair exchange protocols [Aso98] and contain two sub-protocols: an optimistic sub-protocol which is executed between untrusted devices, and (if a participant cannot finish his protocol run) a resolve sub-protocol with a designated TTP. The resolve sub-protocol either nullifies any outstanding commitments (an abort) or completes the exchange (recovery). In Nuovo DRM, if neither party terminates successfully, nothing is exchanged and no outstanding commitments are left. Hence, no particular “abort” protocol is necessary. The C2C exchange thus consists of an optimistic fair exchange protocol and a recovery protocol.

Optimistic fair exchange sub-protocol. The C2C protocol uses the content provider P as the TTP. Figure 7.2 depicts the optimistic exchange sub-protocol.

-
1. $owner(d1) \rightarrow d1 :$ $d2, h(c), r'$
 2. $d1 \rightarrow d2 :$ $d1, n_{d1}$
 3. $d2 \rightarrow d1 :$ $\{n'_{d2}, n_{d1}, d1\}_{sk(d2)}$
 4. $d1 \rightarrow d2 :$ $\{n_{d1}, n'_{d2}, h(c), r', d2\}_{sk(d1)}$
 5. $d2 \rightarrow d1 :$ $\{c\}_{K'}, \{K'\}_{pk(d1)}, \{r', n_{d1}\}_{sk(d2)}$
-

Figure 7.2: C2C exchange: $d1$ buying content from $d2$

In step 4, the purchasing device $d1$ commits to the exchange by sending a signed message to the selling device $d2$. During step 5, $d2$ updates the right associated with c (reflecting that some part of $R_{d2}(c)$ has been used for reselling c) and stores the payment order signed by $d1$ in an atomic action. Note that the atomicity of these actions is necessary to guarantee that $d2$ does not store the payment order without simultaneously updating the right $R_{d2}(c)$.

In this protocol, message 5 may not arrive. For example, due to a malicious $owner(d2)$ that aborts the protocol, or due to a hardware failure. In such cases, $d1$ has already committed to the exchange via the message of step 4, but did not receive the content. To prevent such unfair situations for $d1$, we provide a recovery mechanism to obtain the lost content.

Recovery sub-protocol. The goal of the recovery sub-protocol (Figure 7.3) is to bring a compliant device $d1$, which committed to a C2C exchange (message 4 of the C2C exchange) but did not receive the content it requested (message 5), back to a fair state. Device $d1$ can start a recovery session with the content provider P at any time after sending message 4 in the C2C protocol. The device does so by taking

action $resolves(d1)$ instead of receiving the content at step 5. If a connection with the provider is not available, $d1$ saves the current state and simply waits until it becomes available. Once the recovery protocol has been initiated, $d1$ ignores any further messages from the optimistic run of C2C. The purpose of the recovery is to ensure that $d1$ receives content c and rights r' that $owner(d1)$ wanted (and ostensibly paid for).

$5^r. \quad d1 : resolves(d1)$
 $6^r. \quad d1 \rightarrow P : d1, n'_{d1}$
 $7^r. \quad P \rightarrow d1 : \{n'_P, n'_{d1}, d1\}_{sk(P)}$
 $8^r. \quad d1 \rightarrow P : \{n'_{d1}, n'_P, \langle n_{d1}, n_{d2}, h(c), r', d2 \rangle, r'', P\}_{sk(d1)}$
 $9^r. \quad P \rightarrow d1 : \{c\}_{K''}, \{K''\}_{pk(d1)}, \{r'', n'_{d1}\}_{SK(P)}$

Figure 7.3: C2C recovery: $d1$ recovering a failed exchange

In the recovery protocol, $d1$ and P behave as if $d1$ is purchasing the content-rights bundle (c, r'') from P using the P2C protocol, except that, in message 8^r , $d1$ reports the failed C2C exchange it had with $d1$.

The way P resolves payments of failed exchanges is discussed in more detail in Section 7.7.1. Note that while payment details fall beyond the scope of our formal analysis, the recovery protocol does not.

One can argue that the recovery sub-protocol may also fail due to lossy communication channels. As a way to mitigate this, persistent communication channels for content providers can be built, e.g., using an FTP server as an intermediary. The provider would upload the content, and the device would download it from the server. In order to guarantee fairness, such resilient communication channels are generally unavoidable [Aso98] (c.f. assumption A2).

As a final note, we emphasise that only trusted devices are considered here (assumption A1). These protocols can be trivially attacked if the devices are tampered with (e.g. a corrupted $d1$ would be able to initiate a recovery protocol even after a successful exchange). Methods for detecting circumvented devices and resisting systematic content pirating are described in Section 7.7.

7.5 Formalisation of Nuovo DRM

In this section we describe the steps followed to formally verify that Nuovo DRM achieves its design goals. Our formal approach is based on finite-state model checking [CGP00], which (usually) requires negligible human intervention and, moreover, produces concrete counterexamples, i.e. attack traces, if the design fails to satisfy a desired property. It can therefore be effectively integrated into the design phase. However, a complete security proof of the system cannot, in general, be established by model checking. For an overview on formal methods for verifying security protocols see [Mea03]. As we base our approach on finite-state model-checking, our formal verification must have a finite number of states and thus necessarily concerns a limited instance of the system. Our formal verification can be seen as a sequence of

steps: first, we specify the protocol and the intruder model in the μCRL process algebraic language and generate the corresponding model using the μCRL tool set (version 2.17.12). Second, we state the desired properties in the regular (alternation-free) μ -calculus, and, finally, check the protocol model with regard to the properties in the CADP tool set. These steps are described in detail below.

To highlight important processing steps in the protocols, we now introduce several *abstract* actions. These are used in the formalisation process to define desired behaviours of the protocol.

request($d1, h(c), r, P$) Executed at step 4 in both the P2C and (with appropriate parameters) the C2C protocols by the receiving device, this action indicates the start of the exchange from the receiving device's point of view.

paid($P, h(c), r, d1$) Executed at step 5 of the P2C protocol by P , this action indicates reception of the payment order and sending of content to $d1$.

update($d1, h(c), r, P$) Executed after accepting the message of step 5 of the P2C protocol by $d1$, this action indicates the successful termination of the exchange from $d1$'s point of view.

request($d2, h(c), r', P$) Executed at step 8^r of the C2C recovery protocol by $d2$.

paid($P, h(c), r', d2$) Executed at step 9^r of the C2C recovery protocol by P .

update($d2, h(c), r', P$) Executed after acceptance of the message of step 9^r of the C2C recovery protocol by $d2$.

7.5.1 Modelling language

The complex structure of Nuovo DRM calls for an expressive specification language. We have formalised the Nuovo DRM scheme in μCRL , a language for specifying and verifying distributed systems and protocols in an algebraic style [GP95]. The formalisation is available online¹. A μCRL specification describes a labelled transition system (LTS), in which states represent process terms and edges are labelled with actions. The μCRL tool set [BFG⁺01, BCL⁺07], together with CADP [FGK⁺96] which acts as its back-end, features visualisation, simulation, symbolic reduction, (distributed) state space generation and reduction, model checking and theorem proving capabilities.

We model a security protocol as the asynchronous composition of a finite number of non-deterministic named processes. These processes model roles of participants in the protocol. Processes communicate by sending and receiving messages. A message is a tuple $m = (p; c; q)$, where c is the content of the message and p and q are the identities of the apparent sender and intended receiver process, respectively (this allows the network to route the message to its destination). In denoting communication activity of a message, we omit the active process. That is, sending of message $m = p; c; q$ is denoted as sender p performing the action **send**($q; c$), while receiving this message is denoted as receiver q performing the action **recv**($p; c$). Apart from **send** and **recv**, all other actions of processes are assumed internal, i.e. not communicating with other

¹<http://www.cs.vu.nl/paradiso/formal.php>

$$p ::= a \in \mathcal{A} \mid pa \cdot pb \mid pa + pb \mid pa \triangleleft b \triangleright pb \mid \delta.$$

Figure 7.4: Abbreviated syntax of μCRL process terms

participants. These actions typically symbolise security claims of protocol participants (e.g. *update* in Section 7.4.3). Below, we provide a brief introduction to μCRL , which suffices to understand the formal protocols.

μCRL syntax

In a μCRL specification, processes are represented by process terms, which describe the order in which the actions may happen in a process. A process term p consists of action names and recursion variables (in Figure 7.4, these names and variables are from the set \mathcal{A}) combined by process algebraic operators. The operators ‘ \cdot ’ and ‘ $+$ ’ are used for the sequential and alternative composition (“choice”) of processes pa and pb , respectively. The process expression $pa \triangleleft b \triangleright pb$, where b is a term of type **Bool**, behaves like process pa if b is true, and like process pb if b is false. The predefined action δ represents a deadlock, i.e. from then on, no action can be performed. The process $\sum_{d \in \Delta} P(d)$, where Δ is a (possibly infinite) data domain, behaves as $P(d_1) + P(d_2) + \dots$.

7.5.2 System model

In this section, we show the specification a device buying content in the P2C protocol, the specification of a device buying content in the C2C protocol and the specification of a device recovering a failed C2C exchange. The remaining specifications are the communicating counterparts to these, and thus omitted.

$$\begin{aligned} \mathbf{d1}(\Omega, n_{d1}) = & \\ & \sum_{\substack{r \in Rgts \\ c \in Cont}} \mathbf{recv}(\text{owner}(d1); P, h(c), r) \cdot \mathbf{send}(P; d1, n_{d1}) \cdot \\ & \sum_{n \in Nonce} \mathbf{recv}(P; \{n, n_{d1}, d1\}_{sk(P)}) \cdot \\ & \quad \mathbf{send}(P; \{n_{d1}, n, h(c), r, P\}_{sk(d1)}) \cdot \text{request}(d1, h(c), r, P) \cdot \\ & \quad \sum_{K \in Key} \mathbf{recv}(P; \{c\}_K, \{K\}_{pk(d1)}, \{r, n_{d1}\}_{sk(P)}) \cdot \\ & \quad \text{update}(d1, h(c), r, P) \cdot \mathbf{d1}(\Omega \cup \{\langle c, r \rangle\}, \text{next}(n_{d1})) \end{aligned}$$

Figure 7.5: Nuovo P2C: device $d1$ buying content from provider P

In the μCRL specification of Figure 7.5, we specify the role of the compliant device in the P2C protocol of the Nuovo DRM scheme.

In this specification, $Rgts$, $Nonce$ and Key represent the finite set of rights, nonces and keys available in the protocol, respectively. The set Ω is $d1$'s local collection of content-right bundles, n_{d1} denotes the nonce that is available to $d1$ in the current protocol round, and the function $nxt: Nonce \rightarrow Nonce$, generates a fresh random nonce, given a seed. Note that any discrepancy in the received content is automatically detected in this code: in the last message, if the first part does not agree with the initial $h(c)$, the message will not be accepted.

The specification of Figure 7.5 is the formalisation of the communication of device $d1$ in Figure 7.1. The μCRL specification of the provider's role in the P2C exchange is the logical counterpart of this communication. As such, we omit it here.

$$\begin{aligned}
\mathbf{d1}(\Omega, n_{d1}) = & \\
& \sum_{\substack{r \in Rgts \\ c \in Cont}} \mathbf{recv}(\text{owner}(d1); d2, h(c), r) \cdot \mathbf{send}(d2; d1, n_{d1}) \cdot \\
& \sum_{n \in Nonce} \mathbf{recv}(d2; \{n, n_{d1}, d1\}_{sk(d2)}) \cdot \mathbf{send}(d2; \{n_{d1}, n, h(c), r, d2\}_{sk(d1)}) \cdot \\
& \quad \text{request}(d1, h(c), r, d2) \cdot \\
& \quad \sum_{K \in Key} \mathbf{recv}(d2; \{c\}_K, \{K\}_{pk(d1)}, \{r, n_{d1}\}_{sk(d2)}) \cdot \\
& \quad \text{update}(d1, h(c), r, d2) \cdot \mathbf{d1}(\Omega \cup \{c, r\}, nxt(n_{d1})) \\
& + \mathbf{RESOLVE}(\Omega, n_{d1}, (n_{d1}, n, h(c), r, d2))
\end{aligned}$$

Figure 7.6: Nuovo C2C: device $d1$ buying content from device $d2$

In the μCRL specification in Figure 7.6, we detail the part of the client process that handles buying content in the C2C protocol. This part of the client process is similar to the P2C exchange, up to the *request* abstract action. Then, the process can continue in two different ways. Either the request is fulfilled, and the process continues similar to the P2C protocol, or the process initiates the recovery protocol to resolve a failed exchange. The recovery protocol is fully detailed in Figure 7.7. As such, it is represented by $\mathbf{RESOLVE}(\Omega, n_{d1}, (n_{d1}, n, h(c), r, d2))$ in Figure 7.6.

In case the C2C exchange fails *after* the request action has been executed, the requesting device will execute the recovery protocol from Figure 7.3, which is modelled in Figure 7.7. The recovery protocol largely resembles the P2C protocol, except that the failed C2C exchange is reported. A new nonce n' is generated for this new exchange, and the resolving device reports the failed exchange (by sending $(n_{d1}, n, h(c), r, d2)$ as part of the request for content). Note that under the trusted devices assumption (A1), no device will lie to the provider about a failed exchange.

In formalising the recovery protocol, we chose to have the provider recover using the originally requested rights r . As such, there is no counterpart in the formalisation to r'' from Figure 7.3.

$$\begin{aligned}
& \text{RESOLVE}(\Omega, n_{d1}, (n_{d1}, n, h(c), r, d2)) = \text{resolves}(d1) \cdot \\
& \sum_{\substack{r \in Rgts \\ c \in Cont}} \text{send}(P; d1, n_{d1}) \cdot \\
& \sum_{n' \in Nonce} \text{recv}(P; \{n', n_{d1}, d1\}_{sk(P)}) \cdot \\
& \text{send}(P; \{n_{d1}, n', (n_{d1}, n, h(c), r, d2), P\}_{sk(d1)}) \cdot \\
& \text{request}(d1, h(c), r, P) \cdot \\
& \sum_{K \in Key} \text{recv}(P; \{c\}_K, \{K\}_{pk(d1)}, \{r, n_{d1}\}_{sk(P)}) \cdot \\
& \text{update}(d1, c, r, P) \cdot d1(\Omega \cup \{\langle c, r \rangle\}, \text{next}(n_{d1}))
\end{aligned}$$

Figure 7.7: Nuovo recovery: device $d1$ recovering with provider P

Communication models

We consider two different communication models. The first is a synchronous communication model that is used to verify the effectiveness property (goal G1). No intruder is present in this model and all participants are honest. A process p can send a message m to q only if q can at the same time receive it from p . The synchronisation between these is denoted **com**(p, m, q), which formalises the “ $p \rightarrow q: m$ ” notation of Sections 7.3 and 7.4.

The second model is an asynchronous communication model used to verify the properties G2–G4. In this model, the intruder has complete control over the communication media. When a process p sends a message m with the intention that it should be received by q , it is in fact the intruder that receives it, and it is only from the intruder that q may receive m . The communications between participants of a protocol, via the intruder, are thus asynchronous and, moreover, a participant has no guarantees about the origins of the messages it receives.

Intruder model

We follow the approach of Dolev and Yao to model the intruder [DY83], with some deviations as described below. The Dolev-Yao (DY) intruder has complete control over the network: it intercepts and remembers all transmitted messages, it can encrypt, decrypt and sign messages if it knows the corresponding keys, it can compose and send new messages using its knowledge and can remove or delay messages in favour of others being communicated. As it has complete control over communication media, we assume it plays the role of the communication media. All messages are thus channelled through the intruder. Under the perfect cryptography assumption, this intruder has been shown to be the most powerful attacker model [Cer03]. In our formalisation, this intruder is a non-deterministic process that exhausts all possible sequences of actions, resulting in an LTS which can subsequently be formally checked. Note that the intruder is not necessarily an outside party: it may be a legitimate, though malicious, player in the protocol.

The intruder model used here is different from the DY intruder in two main aspects. These differences stem from the characteristics of the DRM scheme and its requirements:

- Trusted devices significantly limit the power of the intruder. The formalisation ignores the possibility of tampering trusted devices (per assumption A1, mitigation strategies are discussed in Section 7.7). The intruder can, however, deliberately turn off its (otherwise trusted) devices. This is reflected in our model by allowing intruder-controlled devices to non-deterministically choose between continuing or quitting the protocol at each step, except when performing atomic actions. Therefore, in the model, all non-atomic actions a of the devices operated by the intruder are rewritten with $a + \text{off}$. Note that the intruder cannot violate the atomicity of actions for compliant devices. We verify the protocols in the presence of this enriched intruder model to capture possible security threats posed by these behaviours.
- As the intruder can block communications, liveness properties do not hold in the DY model. In order to achieve fairness, essentially a liveness property (see Section 7.4.1), optimistic fair exchange protocols often rely on a “resilient communication channel” (RCC) assumption, see for example [KMZ02]. The RCC assumption guarantees that all transmitted messages will *eventually* reach their destination, provided a recipient for them exists [Aso98]. The behaviour of our intruder model is limited by the RCC assumption, in that it may not indefinitely block the network. Since the intruder is a non-deterministic process in our model, in order to exclude executions that violate the RCC assumption, we impose a fairness constraint on the resulting LTS. Informally stated, the fairness constraint ensures that each process in the system is given a fair chance to execute (see [CGP00]). To denote communications not required by the RCC assumption, we use the action modifier \diamond on regular communication actions (in actions send^\diamond and com^\diamond). A protocol has to achieve its goals even when executions containing com^\diamond actions are avoided. A formal treatment of these issues is found in [CT06].

As a minor deviation from DY, the intruder process performs the abstract action *revealed* when it gets access to a non-encrypted version of any DRM-protected content, to indicate violation of the secrecy requirement (**G2**). This action is of course not triggered when the intruder merely renders an item using its trusted device, which is a normal behaviour in the system.

In the μCRL specification in Figure 7.8, *Agent* represents the set of all honest participants of the protocol and *Msg* represents the set of all messages. Set X is the intruder’s knowledge set. Set Y models the set of messages that still have to be delivered. The set operators \cup and \setminus have their usual meanings. The set $\text{synth}(X)$ represents the (infinite) set of messages that the intruder is able to synthesise from the messages in set X , e.g. by applying pairing, signing and so on. For a complete description of this model please refer to [CT06].

$$\begin{aligned}
I(X, Y) = & \sum_{\substack{p \in Agent \\ m \in Msg}} \mathbf{recv}(p, m, I) \cdot I(X \cup \{m\}, Y \cup \{m\}) + \\
& \sum_{\substack{p \in Agent \\ m \in Msg}} \mathbf{send}(I, m, p) \cdot I(X, Y \setminus \{m\}) \triangleleft m \in Y \triangleright \delta + \\
& \sum_{\substack{p \in Agent \\ m \in Msg}} \mathbf{send}^\circ(I, m, p) \cdot I(X, Y) \triangleleft m \in \mathit{synth}(X) \setminus Y \triangleright \delta + \\
& \sum_{c \in Cont} \mathit{revealed}(c) \cdot \delta \triangleleft c \in \mathit{synth}(X) \triangleright \delta
\end{aligned}$$

Figure 7.8: Nuovo intruder model

7.5.3 Formalisation of Nuovo's goals

The design goals of Nuovo DRM (G1–G4) are formalised in the regular μ -calculus, which is fully described in [MS03]. This logic covers the Nuovo DRM's design goals in its entirety, both safety and liveness, and naturally incorporates data parameters that are exchanged in the protocols. The alternation-free fragment of the regular μ -calculus can be efficiently model checked [MS03], and all the formulae that we have verified are in this fragment. The relevant part of this logic is presented below (for full details, see [MS03]).

$$\begin{aligned}
\alpha &::= a \in \mathcal{A} \mid \neg \alpha \mid \alpha_1 \wedge \alpha_2 \mid \alpha_1 \vee \alpha_2 \\
\beta &::= \alpha \mid \beta_1 \cdot \beta_2 \mid \beta^* \\
\varphi &::= \mathbf{F} \mid \mathbf{T} \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \langle \beta \rangle \varphi \mid [\beta] \varphi \mid \mu X. \varphi
\end{aligned}$$

Figure 7.9: Partial syntax of regular μ -calculus, adapted from [MS03]

The (relevant part of the) syntax of regular μ -calculus is presented in Figure 7.9. Regular μ -calculus consists of *regular formulae* and *state formulae*. Regular formulae describe sets of traces, and are built upon *action formulae* and the standard regular expression operators. We use ‘.’, ‘ \vee ’, ‘ \neg ’ and ‘ $*$ ’ for concatenation, choice, complement and transitive-reflexive closure, respectively, of regular formulae. The syntax of regular formulae as used in the next sections is ranged over by β in Figure 7.9. Variable a ranges over primitive actions from the set \mathcal{A} , such as **send** and **recv**. In addition to this, the following notation is used: \mathbf{F} denotes no action ($\mathbf{F} = a \wedge \neg a$), \mathbf{T} denotes any action ($\mathbf{T} = \neg \mathbf{F}$), and the wild-card action parameter $-$ represents any parameter of an action (e.g. **com**($-$, $-$, $-$) represents any communication action).

State formulae, which express properties of states, are constructed from propositional variables, standard Boolean operators, the possibility modal operator $\langle \dots \rangle$ (used here in the form $\langle \beta \rangle \mathbf{T}$ to express the existence of an execution of the protocol for which the regular formula β holds), the necessity modal operator $[\dots]$ (used here in the form $[\beta] \mathbf{F}$ to express that, for all executions of the protocol, the regular formula β does not hold) and the minimal and maximal fixed point operators μ and ν . A state satisfies

$\mu X.G$ iff it belongs to the minimal solution of the fixed point equation $X = G(X)$, where G is a state formula and X a set of states. The symbols F and T are also used in state formulae. In state formulae they denote the empty set and the entire state space, respectively. The syntax of state formulae as used in the next sections is ranged over by φ in Figure 7.9.

Goals expressed in μ -calculus

This section describes the requirements stated by Nuovo DRM's goals G1–G4 in the regular μ -calculus. Below, the intent of these goals is captured formally. This serves to familiarise the reader with the core formal expressions used in Section 7.6, where the goals are fully formalised.

G1 (effectiveness). Effectiveness means that each purchase request is inevitably responded to, and each received item is preceded by its payment. The first requirement is encoded by stating that after a request occurs in a trace, a matching *update* action must eventually follow. For a device *d1* requesting content *c* from provider *P* with rights *r*, this is formalised as follows:

$$[T^* \cdot \text{request}(d1, c, r, P)] \mu X. (\langle T \rangle T \wedge [\neg \text{update}(d1, c, r, P)]X)$$

The second requirement is encoded by stating that no *update* action occurs without a preceding, matching *paid* action. For a device *d1* updating with content *c* and rights *r* received from provider *P*, this is formalised as

$$[(\neg \text{paid}(P, c, r, d1))^* \cdot \text{update}(d1, c, r, P)]F$$

G2 (secrecy). G2, and the following goals, are checked in presence of an intruder. Secrecy is achieved when the intruder never acquires unprotected content. As mentioned in Section 7.5, the abstract action *revealed*(*c*) denotes the intruder learning unprotected content *c*. Hence, this action should never occur. This is formalised as

$$[T^* \cdot \text{revealed}(c)]F$$

G3 (resisting content masquerading). Content masquerading occurs when a device accepts a bundle of content and rights it did not request. Non-occurrence of this situation is formalised for device *d1* and content *c*, rights *r* from *P* as

$$[(\neg \text{request}(d1, c, r, P))^* \cdot \text{update}(d1, c, r, P)]F$$

G4. (strong fairness). Strong fairness must hold for each participant. Thus, it breaks down into the following cases for provider *P*, devices *d1*, *d2*, content *c* and rights *r*:

- Strong fairness for *P*.

No compliant device receives protected content, unless the corresponding payment has been made. This means that a device does not update without paying, and that no device can replay an update event. Formalised with respect to device *d1*, rights *r* and content *c*:

$$[(\neg \text{paid}(P, c, r, d1))^* \cdot \text{update}(d1, c, r, P)]F \wedge \\ [T^* \cdot \text{update}(d1, c, r, P) \cdot (\neg \text{paid}(P, c, r, d1))^* \cdot \text{update}(d1, c, r, P)]F$$

- Strong fairness for device $d2$ when buying from provider P .
If device $d1$ pays for content, it will eventually receive it. Expressed for buying content c and rights r from provider P :

$$\begin{aligned} & [\mathbf{T}^* \cdot \text{request}(d2, c, r, P) \cdot (\neg(\text{update}(d2, c, r, P)))^*] \\ & \langle (\neg \text{com}^\diamond(-, -, -))^* \cdot (\text{update}(d2, c, r, P)) \rangle \mathbf{T} \end{aligned}$$

- Strong fairness for device $d2$ when buying from reselling device $d1$.
Buying content c and rights r from a reselling device $d1$ is treated similarly, except that recovery is taken into account. To resolve a failed exchange, a TTP is needed. As there are only finitely many TTP's in the model, the intruder can occupy all of them, preventing other agents from reaching them. Such a denial of service attack can be mitigated in practice (e.g. by placing time limits on transactions with a TTP). Here, we abstract away from timing aspects and use the action last_{ttp} to indicate that the intruder has exhausted all TTPs in the model. As long as this action has not occurred, there is at least one TTP available. Thus, the first part of the expression describes a C2C exchange, and the second part the recovery protocol:

$$\begin{aligned} & [\mathbf{T}^* \cdot \text{request}(d2, c, r, d1) \cdot (\neg(\text{resolves}(d2) \vee \text{update}(d2, c, r, d1)))^*] \\ & \langle (\neg \text{com}^\diamond(-, -, -))^* \cdot (\text{resolves}(d2) \vee \text{update}(d2, c, r, d1)) \rangle \mathbf{T} \\ & \wedge \\ & [(\neg \text{last}_{ttp})^* \cdot \text{request}(d2, c, r, d1) \cdot (\neg \text{last}_{ttp})^* \cdot \text{resolves}(d2) \cdot \\ & (\neg(\text{update}(d2, c, r, P) \vee \text{last}_{ttp}))^*] \\ & \langle (\neg \text{com}^\diamond(-, -, -))^* \cdot \text{update}(d2, c, r, P) \rangle \mathbf{T} \end{aligned}$$

Note that the strong fairness notion that is formalised and checked here subsumes the timeliness property, because when $d2$ starts the recovery protocol (which it can autonomously do), it always recovers to a fair state without any help from $d1$.

- Strong fairness for device $d2$ when reselling content to device $d1$.
No compliant device receives content, without a corresponding payment. This case is equivalent to fairness for the provider.

7.6 Analysis results

In this section we describe the results obtained from the formal analysis of the Nuovo DRM scheme. Our analysis has the following properties: the intruder is allowed to have access to unbounded resources of data (like fresh nonces), should it need them to exploit the protocol. We consider only a finite number of concurrent sessions of the protocol, i.e. each participant is provided a finite number of fresh nonces to start new exchange sessions. Although this does not, in general, constitute a proof of security for a protocol, in many practical situations it suffices. As security of cryptographic protocols is not decidable (e.g. see [CS02]), a trade-off has to be made between completeness of the proofs and their automation. Our analysis method is fully automatic, evaluating specific use cases in specific settings (as detailed below). Following [DY83], we assume perfect cryptography and do not consider attacks resulting from weaknesses of the cryptographic primitives used in protocols. Type-flaw attacks (attacks based on confusion of the type of a specific field in a message) are also omitted from our analysis. Such attacks can be prevented [HLS00].

Our formal analysis consists of two scenarios. A scenario explicitly describes one specific execution of the protocol, with specific actors, specific assumptions and a specific intruder model. The first scenario verifies effectiveness (G1) while using the synchronous communication model of Section 7.5.2, in absence of an intruder. The second scenario uses the asynchronous communication model of Section 7.5.2 and the modified DY intruder model of Section 7.5.2 to verify the remaining properties (G2–G4). Both scenarios operate under the assumptions of Section 7.4.2. Both scenarios describe a setting with two compliant devices $d1$ and $d2$, three different pieces of content and two different rights, the first right allowing to resell the second. In the second scenario, the devices are controlled (but not tampered) by the intruder of Section 7.5.2. Below, P , as before, represents the trusted content provider. The formulae in the following results use abstract actions to improve the readability of the proved theorems. These actions are explained in Sections 7.4.3 and 7.5.2. A complete formalisation of these actions can be found in [JKT06].

We require that Nuovo satisfies its design goals for these scenarios.

Lemma 6. *Nuovo DRM achieves design goals G1–G4 in scenarios S_0 and S_1 .*

Note that effectiveness (G1) is only applicable in scenario S_0 , and that as scenario S_0 does not have an intruder, it suffices to verify that goals G2–G4 hold in scenario S_1 . Below, we set about proving this lemma.

7.6.1 Honest scenario

Scenario S_0 describes the interaction between one provider P and two devices ($d1$ and $d2$). The communication network is assumed operational and no malicious agent is present. The scenario runs as follows: device $d1$ is ordered to buy an item and reselling rights from P . Then, $d1$ resells the purchased item to $d2$. As this scenario is only intended as a sanity check for the protocol, we believe correct behaviour in this scenario with two devices running multiple instances of the protocol is strong supporting evidence for that in general. For that reason, as well as the increased computational load of having more devices, we limited this scenario to only two devices. The scenario was checked using the EVALUATOR 3.0 model checker from the CADP tool set, confirming that it is deadlock-free, and effective as specified below.

Result 1. *Nuovo DRM is effective for scenario S_0 , meaning that it satisfies the following properties:*

1. *Each purchase request is inevitably responded.*

$$\begin{aligned} &\forall c \in \text{Cont}, r \in \text{Rgts}. \\ &[\mathbf{T}^* \cdot \text{request}(d1, c, r, P)] \mu X. ((\mathbf{T})\mathbf{T} \wedge [\neg \text{update}(d1, c, r, P)]X) \quad \wedge \\ &[\mathbf{T}^* \cdot \text{request}(d2, c, r, d1)] \mu X. ((\mathbf{T})\mathbf{T} \wedge [\neg \text{update}(d2, c, r, d1)]X) \end{aligned}$$

2. *Each received item is preceded by its payment.*

$$\begin{aligned} &\forall c \in \text{Cont}, r \in \text{Rgts}. [(\neg \text{paid}(P, c, r, d1))^* \cdot \text{update}(d1, c, r, P)]\mathbf{F} \quad \wedge \\ &[(\neg \text{paid}(d1, c, r, d2))^* \cdot \text{update}(d2, c, r, d1)]\mathbf{F} \end{aligned}$$

7.6.2 Dishonest scenario

Scenario S_1 describes the interaction of an intruder I , two compliant devices $d1$ and $d2$, and three providers. The intruder controls the communication network and is the owner of devices $d1$ and $d2$. The intruder can instruct the compliant devices to purchase items and rights from the provider P , exchange items between themselves and resolve a pending transaction. Moreover, the compliant device $d1$ can non-deterministically choose between following or aborting the protocol at each step, which models the ability of the intruder to turn the device off (see Section 7.5.2, “intruder model”). We model three concurrent runs of the content provider P , and three sequential runs of each of $d1$ and $d2$. Although this is again a limited setting, the intruder capabilities are as strong as in a setting with more devices. Adding more devices would increase the number of honest users, not increase the capabilities of the intruder. This reasoning coupled with the computational power necessary to handle the generation of the state space directed us to limit the scenario thusly. The resulting model was checked using the EVALUATOR 3.0 model checker from the CADP tool set and the following results were proven.

Result 2. *Nuovo DRM provides secrecy in scenario S_1 , i.e. no protected content is revealed to the intruder.*

$$\forall c \in \text{Cont. } [\mathbf{T}^* \cdot \text{revealed}(c)]\mathbf{F}$$

Result 3. *Nuovo DRM resists content masquerading attacks in S_1 , ensuring that a compliant device only receives the content which it has requested.*

$$\begin{aligned} \forall c \in \text{Cont}, r \in \text{Rgts. } & [(\neg \text{request}(d1, c, r, d2))^* \cdot \text{update}(d1, c, r, d2)]\mathbf{F} \wedge \\ & [(\neg \text{request}(d2, c, r, d1))^* \cdot \text{update}(d2, c, r, d1)]\mathbf{F} \wedge \\ & [(\neg \text{request}(d1, c, r, P))^* \cdot \text{update}(a, c, r, P)]\mathbf{F} \wedge \\ & [(\neg \text{request}(d2, c, r, P))^* \cdot \text{update}(a, c, r, P)]\mathbf{F}. \end{aligned}$$

In addition, the intruder cannot feed the self-fabricated content c_0 to compliant devices:

$$\begin{aligned} \forall r \in \text{Rgts. } & [\mathbf{T}^* \cdot \text{update}(d1, c_0, r, d2)]\mathbf{F} \wedge \\ & [\mathbf{T}^* \cdot \text{update}(d2, c_0, r, d1)]\mathbf{F} \wedge \\ & [\mathbf{T}^* \cdot \text{update}(d1, c_0, r, P)]\mathbf{F} \wedge \\ & [\mathbf{T}^* \cdot \text{update}(d2, c_0, r, P)]\mathbf{F}. \end{aligned}$$

Result 4. *Nuovo DRM provides strong fairness in S_1 for P , i.e. no compliant device receives protected content, unless the corresponding payment has been made to P .*

$$\begin{aligned} \forall a \in \{d1, d2\}, c \in \text{Cont}, r \in \text{Rgts. } & [(\neg \text{paid}(P, c, r, d1))^* \cdot \text{update}(d1, c, r, P)]\mathbf{F} \\ & \wedge [(\neg \text{paid}(P, c, r, d2))^* \cdot \text{update}(d2, c, r, P)]\mathbf{F} \\ & \wedge [\mathbf{T}^* \cdot \text{update}(d1, c, r, P) \cdot (\neg \text{paid}(P, c, r, d1))^* \cdot \text{update}(d1, c, r, P)]\mathbf{F} \\ & \wedge [\mathbf{T}^* \cdot \text{update}(d2, c, r, P) \cdot (\neg \text{paid}(P, c, r, d2))^* \cdot \text{update}(d2, c, r, P)]\mathbf{F} \end{aligned}$$

Result 5. *Nuovo DRM provides strong fairness in S_1 for $d2$, as formalised below. (Note that strong fairness for $d1$ is not guaranteed here, as it can abort the protocol prematurely. A protocol guarantees security only for the participants that follow the protocol.)*

1. *As a client: if a compliant device pays (a provider or reseller device) for a content, it will eventually receive it.*

$$\begin{aligned}
& \forall c \in \text{Cont}, r \in \text{Rgts}. [\mathsf{T}^* \cdot \text{request}(d2, c, r, P) \cdot (\neg(\text{update}(d2, c, r, P)))^*] \\
& \quad \langle (\neg \text{com}^\diamond(-, -, -))^* \cdot (\text{update}(d2, c, r, P)) \rangle \mathsf{T} \\
& \quad \wedge \\
& \forall c \in \text{Cont}, r \in \text{Rgts}. \\
& \quad [\mathsf{T}^* \cdot \text{request}(d2, c, r, d1) \cdot (\neg(\text{resolves}(d2) \vee \text{update}(d2, c, r, d1)))^*] \\
& \quad \langle (\neg \text{com}^\diamond(-, -, -))^* \cdot (\text{resolves}(d2) \vee \text{update}(d2, c, r, d1)) \rangle \mathsf{T} \wedge \\
& \quad [(\neg \text{last}_{\text{ttp}})^* \cdot \text{request}(d2, c, r, d1) \cdot (\neg \text{last}_{\text{ttp}})^* \cdot \text{resolves}(d2) \cdot \\
& \quad \quad (\neg(\text{update}(d2, c, r, P) \vee \text{last}_{\text{ttp}}))^*] \\
& \quad \langle (\neg \text{com}^\diamond(-, -, -))^* \cdot \text{update}(d2, c, r, P) \rangle \mathsf{T}
\end{aligned}$$

2. *As a reseller: no compliant device receives a content from a reseller device, unless the corresponding payment has already been made to the reseller.*

$$\begin{aligned}
& \forall c \in \text{Cont}, r \in \text{Rgts}. \\
& \quad [(\neg \text{paid}(d2, c, r, d1))^* \cdot \text{update}(d1, c, r, d2)] \mathsf{F} \wedge \\
& \quad [\mathsf{T}^* \cdot \text{update}(d1, c, r, d2) \cdot (\neg \text{paid}(d2, c, r, d1))^* \cdot \text{update}(d1, c, r, d2)] \mathsf{F}
\end{aligned}$$

The proof of Lemma 6 is based on the above Results 1–5. We now prove Lemma 6 using Results 1-5.

Proof. (Lemma 6)

- G1 is achieved based on Result 1;
- Result 2 implies G2;
- Result 3 guarantees achieving G3;
- Results 4 and 5 guarantee G4.

□

Note that Lemma 6 does not prove that Nuovo DRM achieves its design goals in *all* possible scenarios. It does support and provide credence for this statement.

This lemma underpins the security of Nuovo DRM from a formal point of view. The next section outlines procedures to ensure that use of Nuovo DRM in practice will remain close to scenario's S_0 and S_1 .

7.7 Nuovo DRM mitigation procedures

To fully address the research question, Nuovo DRM must address security from a practical stance as well as from a formal point of view. Nuovo DRM's security, as formally analysed above, is based on assumptions A1–A4. However, practical DRM systems such as iTunes (by Apple) and Windows Media DRM (by Microsoft) have illustrated time and again that DRM systems cannot rely on such assumptions, but must provide mitigation strategies for when the system's security is breached. In this section, mitigation procedures for Nuovo DRM are introduced and discussed. As noted in Section 7.4.2 and explained in the discussion of the intruder model in

Section 7.5.2, assumptions A1 (tamper-proof devices) and A2 (resilient communications) limit the power of the intruder. The mitigation procedures thus should address violations of these assumptions.

This section begins with a procedure addressing non-resilient communications in peer-to-peer exchanges. The procedure details how the provider resolves failed peer-to-peer exchanges. Next, we focus on non-compliant devices. A tampered device violates assumption A1, and may access all its content without respecting the associated rights. There is no way to prevent content being accessed on the compromised device. However, interactions between compliant and non-compliant devices can be minimised. We describe two procedures to support this: one procedure to detect compromised devices and one procedure to minimise interactions between compliant and detected, compromised devices.

7.7.1 Resolving C2C disputes at the TTP

We define $\text{price}: Rgts \rightarrow \mathbb{N}$. Given a $r \in Rgts$, $\text{price}(r)$ denotes the price that has been assigned to r (see assumption A4). In the recovery protocol, the provider will agree to resolve a failed C2C exchange for right r'' (steps $8^r, 9^r$) iff $\text{price}(r'') \geq \text{price}(r')$ (from step 8^r); below we see why this condition is necessary. In line with assumption A1, we only consider compliant devices that need to recover – the device cannot lie about $\text{price}(r')$ and $\text{price}(r'')$. In practice, resellers will usually propose prices which are lower than the main vendor's price for buying that single item, hence automatically satisfying this requirement.

We require P to maintain a persistent log of the resolved disputes. Assume that $d2$ tries to recover from an unsuccessful exchange with $d1$. As a result of the atomicity of $d1$'s actions in the optimistic sub-protocol, only the following situations are possible: Either $d1$ has updated $R_{d1}(c)$ and has the payment order of message 4 of C2C (which it is thus entitled to have), or $d1$ does not have the payment order and has not updated $R_{d1}(c)$. In the latter case, the combination of the failed optimistic protocol and its subsequent resolution simply boils down to a P2C exchange. If $d1$ owns the payment order from $d2$, two different cases are possible:

1. If $d2$ recovers before $d1$ tries to encash the payment order, P will pay $d1$, as $d2$ has already paid P . Since $\text{price}(r'') > \text{price}(r')$, P can always pay $d1$ its share of the transaction. Therefore, the actual payment to P in this particular exchange sums up to $\text{price}(r'') - \text{price}(r')$. Note that although the net price P is paid for selling c to $d2$ in this particular exchange is too low, it is indeed fair because P has already been paid by $d1$ when $d1$ bought the right to resell c .
2. If $d2$ recovers after $d1$ has encashed the payment order, P will not charge $d2$, because $d2$ has already paid the price to $d1$ and $d1$ has updated the right $R_{d1}(c)$, for which it has already (directly or indirectly) paid P .

Note that $d1$ and $d2$ cannot collude to cheat P by $d1$ offering item c to $d2$ for an extremely low price and then resolving the request to P . To make this clear, consider the following use case:

- Cost of buying song c for playing only, directly from $P = \$1.00$.

- Cost of buying song c for 50 resell rights from $P = \$0.80 \times 50 = \40 .
- Cost of buying song c for playing only, from reseller $d1 = \$0.90$.

First, this scenario describes a viable business model: $d2$ would rather buy c from $d1$ than directly from P because of the \$0.10 difference. $d1$ has incentive to act as reseller since it can make a profit of $\$(0.9-0.8) \times 50$ if all the songs are sold. P would rather make one sale of 50 rights to $d1$ than sell to 50 $d2$ s directly, to avoid all sorts of administration, processing and other per-transaction costs (it is common for services such as MasterCard or PayPal to have per-transaction charges consisting of a fixed part and a part dependent to the transaction).

If $d1$ offers c to $d2$ for \$0.01 and they resolve it to P , P would transfer the money from $d2$'s account to $d1$'s without being paid in this exchange (P would accept resolving such unnecessary disputes to assure its customers that in case of real problems, they can resort to P). However, $d1$ is the one who actually loses money. P 's profit was already made when the resell rights were sold to $d1$, and $d2$ has exploited a very good offer on c . If this scenario is repeated enough, $d1$ will sell contents for $\$0.01 \times 50 = \0.50 . At the end of the day, $d1$ paid $\$40 - \$1.00 - 0.50 = \$38.50$ more than the market price for c .

Seeing that the TTP cannot be cheated by compliant devices, even if their owners are colluding, the provider can safely be considered a TTP. The provider's interests are not harmed, and the role of TTP allows it to offer an extra service to its customers.

7.7.2 Detection of compromised devices

The security of Nuovo DRM hinges on the compliance of the certified user devices. However, it is reasonable to assume that over time, some of these devices will be compromised. In this section, we examine how to detect compromised devices. As in [NPG⁺05], the proposed mechanism aims at detecting powerful attackers and systematic content pirating, rather than occasionally misbehaving users. Hence, we consider a device to be compromised if it misbehaves frequently. So instead of compliance checks, the aim is to detect devices exhibiting deviant behaviour.

Nuovo DRM enables content redistribution in a controlled manner. In addition to regular attacks on DRM systems, Nuovo DRM has to consider attacks on content redistribution as well. Regular attacks and countermeasures are already discussed in [NPG⁺05]. This leaves attacks on content redistribution. As compliant devices will not misbehave, only compromised devices can perform such attacks. A compromised device can attack content redistribution phase in two ways. First, it can overuse a right to resell content; secondly, it can try to avoid paying for content it receives (by not having sufficient funds). These are discussed in order.

Fund exchange during content redistribution is clearly a crucial point of attack. Successful attacks would undermine users' confidence in the system and the benefit to the attacker is clear: acquire more funds and spend less funds, respectively. In order to address this, we introduce the following assumptions (for compliant devices) on funds, which so far have not been considered.

A5 (device funds). When a compliant device signs a payment order, the payment

order is cash-able. This can be accomplished, for instance, by providing each compliant device with some credit, which can be spent and recharged.

A6 (traceability of funds). The banking system (responsible for encashing payment orders) cooperates with content providers to catch malevolent users. For the sake of simplicity, the content provider and the bank are considered as one and the same entity.

Note this assumption may not be universally acceptable, e.g. due to geographical diversity of content providers and banking systems used by customers. Nevertheless, the required degree of collaboration makes the assumption practically tenable.

First, consider an overuse of reselling rights. To detect large scale overselling, the provider reconstructs the chain of sold rights. This is possible because of assumption A6 – to acquire payment for sold rights, devices need to contact the provider.

To this end, the provider maintains a directed weighted graph $G = (V, E)$ for each sold content-right combination, that may be resold. Each node $v \in V$ represents a device and the weighted edges of the graph ($E: V \times V \rightarrow \mathbb{N}$) represents right transfers between two compliant devices. For each $v \in V$, *weight difference* is the difference between outgoing weight and incoming weight. Formally:

$$\Delta(v) = \sum_{v' \in V} E(v, v') - \sum_{v' \in V} E(v', v) \quad (7.1)$$

Let $U \subseteq V$ be the set of nodes that have sold a content-rights bundle, but have not yet encashed the payment order. If v_c is a compromised device which engages in large scale overselling, after a reasonable amount of time, the provider will detect v_c 's behaviour by noting that the weight difference of v_c plus the number of yet-to-cash rights are positive, i.e.

$$\Delta(v_c) + \sum_{u \in U} \Delta(u) > 0 \quad (7.2)$$

By putting time limits on encashing payment orders, a provider can control the time bound on detecting compromised devices. Such an approach requires the payment orders to be timestamped by the device issuing the order. Timestamps of compliant devices can be trusted, and the overhead to check the timestamp against the time limit is very small. Hence this solution scales well with the size of the system.

Secondly, a compromised device can refuse to pay for the content it receives. According to assumption A5, user devices are provided with (and thus aware of) credits. Therefore, the second attack could easily be detected by the banking entity (collaborating with the providers) when a device signs a payment order without having enough credit for that, as a compliant device would not cause this error.

7.7.3 Isolating compromised devices

Given that cheating – and thus compromised – devices are detected, countermeasures can be taken. Confiscation of the compromised device is of course preferred. However, in practice this will not always be possible. Instead, a Device Revocation List (DRL), containing public keys of detected compromised devices, can be used to limit interactions of compliant devices with them. To ensure correct working of device revocation, soundness of this DRL is required – no compliant device is ever listed on the DRL.

Completeness of the DRL also seems desirable: all compromised devices are listed on the DRL. However, as more and more compromised devices are detected, such a list could grow quite large over time. Given that not all devices are equally likely to interact, there is a trade-off between effectiveness and size of the DRL stored on a compliant device.

Considering these two properties immediately gives rise to two alternative ways of distributing the DRL: optimising effectiveness and optimising size, respectively. Optimising effectiveness of the DRL is done by keeping the complete DRL and updating it at all possible opportunities. In this case, each device has a *complete copy* of the DRL. Optimising size of the DRL is done by adding only those compromised devices with which a device has had contact with earlier – only *check neighbours*.

Below we examine these two distribution schemes, and propose and refine variants that aim to balance these two considerations. Furthermore, estimates for the effectiveness and size of the per-device stored DRL are established. The following notation is used below:

drl The main DRL, as kept by P

drl_{d1} the DRL as kept by device $d1$

$ghbr_{d1}$ the list of devices with which device $d1$ has had contact. To keep the size of this list within reasonable bounds, it is reset after each contact with the provider that updates the DRL.

Updates of variables are denoted as $a', b', c' := a, b, c$, where the left-hand side denotes the variables a, b, c after the update and the right-hand side expresses the current values. Contact between two parties a and b is denoted as $a \leftrightarrow b$, irrespective of which contact is initiated (P2C, C2C, recovery).

complete copy: Each device keeps a copy of the entire DRL. Updates:

- on $d1 \leftrightarrow P$: $drl'_{d1} := drl$.
- on $d1 \leftrightarrow d2$: $drl'_{d1}, ghbr'_{d1} := drl_{d1} \cup drl_{d2}, ghbr_{d1} \cup \{d2\}$.

neighbour-check: A device only lists those revoked devices, with which it has had contact. Updates:

- on $d1 \leftrightarrow P$: $drl'_{d1} := ghbr_{d1} \cap drl$.
- on $d1 \leftrightarrow d2$: $ghbr'_{d1} := ghbr_{d1} \cup \{d2\}$.

The next scheme strikes a balance between the two approaches above. On contacting the provider, it updates as *neighbour-check* (remembering the old list, however). On contact with another device, it updates as *complete copy*.

propagated list: Each device includes the DRL of all devices it has contacted. Updates:

- on $d1 \leftrightarrow P$: $drl'_{d1} := (drl_{d1} \cup neighr_{d1}) \cap drl$.
- on $d1 \leftrightarrow d2$: $drl'_{d1}, neighr'_{d1} := drl_{d1} \cup drl_{d2}, neighr_{d1} \cup \{d2\}$.

Depending on the interconnectivity of devices, the entire DRL could quickly reside on each device (in accordance with the “six degrees of separation” idea, which roughly states that everyone is at most six handshakes away from every other person on the planet). In this case, *propagated list* would be a complex version of *complete copy*. A size-limiting refinement of *propagated list* is to forward not simply all incoming DRLs, but only forward those devices on the DRL with which a device has had contact itself. To this end, the per-device DRL is partitioned in two: $self_{d1}$ lists those revoked devices, with which $d1$ has had contact. $rest_{d1}$ accumulates the DRLs learned from other devices. So $drl_{d1} = self_{d1} \cup rest_{d1}$. With this in mind, *propagated list* is refined as follows.

restricted propagation: each device includes the DRL of all devices it has contacted, but does not propagate this further. Updates:

- on $d1 \leftrightarrow P$: $self'_{d1} := neighr_{d1} \cap drl$.
- on $d1 \leftrightarrow d2$: $rest'_{d1}, neighr'_{d1} := rest_{d1} \cup self_{d2}, neighr_{d1} \cup \{d2\}$.

Note that, given the partitioning of the DRL into two, another approach is to validate the received DRLs before further propagation. Thus, $drl_{d1} = self_{d1}$, which is updated as described below.

validated propagation: propagated lists are only included after validation against the provider’s DRL. Updates:

- on $d1 \leftrightarrow P$: $self'_{d1} := (self_{d1} \cup rest_{d1} \cup neighr_{d1}) \cap drl$.
- on $d1 \leftrightarrow d2$: $rest'_{d1}, neighr'_{d1} := rest_{d1} \cup self_{d2}, neighr_{d1} \cup \{d2\}$.

Remark that validated propagation is a sanitised version of *propagated list*. Hence, this distribution model is not considered further.

Of the above schemes, restricted propagation seems to offer the best trade-off of list length versus usefulness of the DRL. On a further note, the impact of a corrupted DRL can be limited if each device cleans its DRL every time it contacts the provider ($drl_{d1} := drl_{d1} \cap drl$). This would also allow the provider to “un-revoke” devices.

To get a feeling for the inherent effectiveness and list size of a given distribution scheme, the distribution schemes are examined in the following setting.

Use case

Given the number of devices $n = 10^7$ and the fraction of compromised devices $e = 10^{-3}$ (i.e. the number of compromised devices is in the order of 10^4). Assume that compliant devices interact via a (very regular) network structure, each device interacting with k other devices, that no two neighbours interact with each other (thus the number of unique neighbours of second degree is $k \cdot (k - 1)$). Figure 7.10 depicts one node in such a network, where $k = 4$.

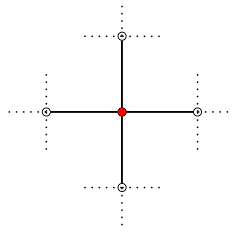


Figure 7.10: One node in a network of $k = 4$ neighbours

Furthermore, assume that the secure memory of a device can hold a list of at most 10 id's (due to memory limitations). Below, we calculate for each distribution scheme how many unique devices a device may interact, to still restrict interactions with compromised devices.

complete copy: Each device needs to store the complete DRL of all compromised devices. This DRL will eventually consist of $e \cdot n = 10^4$ entries. This list cannot be stored in secure memory.

friends-check: Each device needs to store only those compromised devices it encountered, i.e. the list size is $e \cdot k$. For a maximum list size of 10, this distribution scheme can accommodate interconnections of up to $k = 10^4$.

propagated list: After every interaction, the DRL can only grow. Given the extremely regular, totally connected network of the use case, this means that eventually, every revoked device will be listed. The time it takes for this to happen depends on the degree of separation. Within the setting sketched above, the degree of separation is approximately $\frac{\log n}{\log k-1}$.

restricted propagation: The restricted propagation scheme will only revoke compromised devices that interact with a device or with a neighbour of the device. The number of entries in the per-device DRL depends on the number of neighbours k as follows: $e \cdot k \cdot (k - 1)$. For a maximum list size of 10 this means $10^{-3} \cdot (k^2 - k) \leq 10$, which is approximately $k^2 \leq 10^4$. Hence this suffices for interacting with up to 100 different, unique devices.

In order to measure the effectiveness of a distribution method, we consider how limiting a scheme is for a compromised device. This is equal to the number of compliant devices which lists a compromised device – i.e., the number of devices that will not interact with a compromised device. Note that due to our over-approximation that a compromised device (or its owner) is aware of which devices know it is compromised and can contact any other device, compromised devices may be considered to be uniformly distributed throughout the set of devices. Hence, the number of devices not interacting with a specific compromised device is equal to the average size of the per-device DRL. Given the uniformity assumption, this metric is equal to the size of the list, which was discussed above.

7.8 Summary

Vulnerabilities in the NPGCT DRM system were identified and addressed. Besides this, several procedures were introduced for detection and revocation of compromised devices. The extended system, Nuovo DRM, is inherently complicated (as many other DRM schemes are) and, thus, error prone. This calls for expressive and powerful formal verification tools to provide a certain degree of confidence in the security and fairness of the system. We have analysed and validated our design goals on a finite model of Nuovo DRM. This is of course no silver bullet: our formal verification is not complete as it abstracts away many details of the system.

To support a practical implementation of Nuovo DRM, the possibility of compromised devices has to be taken into account. To this end, procedures for both detection and revocation of compromised devices are introduced by Nuovo DRM. The distribution of revocation lists was discussed, and several alternative distribution models were compared.

As future work, we are considering several extensions to the formal analysis. For example, the accountability of the provider, which is taken as non-disputable in this study, can be verified. Additionally, possible privacy concerns of customers and the payment phase can be incorporated into the formal model.

The comparison of the various distribution models for revocation lists (especially the effectiveness of said models) is strongly influenced by the assumed uniformity of the network of connected devices. As future work, we intend to investigate the notion of effectiveness in a less uniform setting.

Part III

Concluding remarks

Conclusions

In this chapter, we review the results that were achieved in this thesis. The main lesson from both parts of the thesis is that we need a structured approach to the study of security.

Our studies of the security of electronic voting and of digital rights management began with an analysis of each domain and its requirements. In both these domains, some requirements interact with each other. This underlines the necessity of understanding the context of a security requirement before studying the requirement itself.

In addition, in voting (Chapter 3) we found that some systems do not achieve security, despite arguments and even proofs of security. To prevent these pitfalls, reasoning on security must be done in a mathematically precise format. The security objectives as well as the intruder capabilities must be included in this format. Only in this way can we truly ensure that a security requirement can be satisfied. However, in DRM (Chapter 7), we noted that a formal approach is based on certain assumptions. In practice, such assumptions can be violated. Technical measures can ensure the desired security, however, it is often not possible to technically enforce the correct use of technical measures. This means that technical measures alone do not suffice to achieve security in a deployed system. They must be supported by physical means such as tamper-proof devices, and by procedural means such as mitigation procedures.

Thus, understanding security requires a structured approach, which begins with an understanding of the context. This is followed by formalising the desired security objectives and the intruder capabilities. This underlies a theoretical understanding of security, but for security requirements to be satisfied in practice, the theoretical approach must be supported by physical and procedural means.

The rest of this chapter provides detailed conclusions for each of the two parts of the thesis.

8.1 Privacy in voting

In Chapter 2, we analysed the domain of voting. From this domain analysis we derived a set of core security requirements for voting systems. Comparison with existing terminology in literature highlighted a confusion surrounding privacy. Accordingly, the following research question was posed.

Research question I. *Investigate and improve understanding of privacy in voting.*

We addressed this question by first investigating the understanding of privacy in Chapter 3. There, we described mechanisms that are used to introduce support for privacy in voting systems. Furthermore, we provided a survey of voting systems focusing on receipt-free and end-to-end verifiable voting systems. In the survey, we saw that several claims of receipt-freeness were proven false by newer works.

This led us to see where understanding of privacy can be improved. In Chapter 4, we set out to provide an improved formalisation of privacy in voting. We took note of the formalisations by Delaune, Kremer, and Ryan and saw that their formalisations are qualitative in nature. Any attack where privacy is not nullified is not captured by their approach. In our view, the only way to detect even the most subtle privacy attacks is to capture every reduction of privacy. Therefore, we designed a formal framework that is able to compute the exact set of candidates a voter may have voted for in a given situation.

We proposed several models of voter conspiracy, that cover the standard communication assumptions found in literature (as found in Chapter 3). The framework views privacy in a quantified manner, which enables to detect receipts which reduce, but not nullify privacy. We discussed how the framework captures and extends the notions of receipt-freeness and coercion-resistance. The chapter ended with a discussion on the relation between privacy and verifiability, which was illustrated by the 3BS system.

We illustrated the application of the concepts and framework in Chapter 5. The main contributions of this chapter is to illustrate how the concepts and framework improve understanding in voting. This was done by analysing two voting systems using the framework, identifying privacy weaknesses in either, and providing suggestions for improvement. Furthermore, the concepts of Chapter 4 were used to understand the trade off between privacy and verifiability inherent in the ThreeBallot voting system.

Thus, by means of a formalisation of the notion of privacy we have achieved the goal of investigating and improving the understanding of privacy in voting.

8.1.1 Future work

There are several directions in which this work can be extended. First of all, application of the framework is a complex process, as evidenced in Chapter 5. An automated approach of this process is a logical next step. Given that the process language was deliberately chosen to remain close to process algebras such as ACP and μ CRL, we are considering an implementation based on μ CRL or mCRL2.

Another issue is the question of decidability. The basis of our knowledge modelling is decidable, as proven by Baskar et al. However, we have yet to investigate whether this holds for the integrated models of the privacy primitives.

Furthermore, while the framework quantifies privacy, it does not take into account various probabilistic aspects of voting systems. For example, the distribution of votes in the result is related to the chance that a voter voted for a candidate in the voter's choice group. Another example are probabilistic aspects of voting systems, such as mixing in mixnets. Here, the framework assumes mixnets work perfectly. However, this assumption can be violated by some mixnets. A mixnet may be biased, in that certain ways of shuffling are more likely than others. A next step is to enrich the

quantitative aspect of the framework with probabilities, so that such a bias is taken into account.

In addition, the increased understanding of privacy is to be used for designing voting systems. There are two directions in which this can be taken. On the one hand, we seek to design voting systems that offer robust privacy, even in the face of strong coercion (type 2c). On the other hand, we are interested in voting systems that explore the inherent trade-off between verifiability and privacy further.

Apart from technical extensions of the framework, we are also interested in extending the concepts of the framework themselves. In the discussion on the 3BS system in Chapter 5, we saw that a coalition of voters has a greater ability to protect itself than an individual voter. A future direction is to further investigate and characterise the defensive abilities of coalitions.

Finally, voting systems provide an illustration of systems where enabling privacy is not sufficient – they must enforce privacy. The notion of enforced privacy can be carried over to other domains, such as electronic health care records or online auctions.

8.2 Fairness in digital exchange

In the second part of the thesis, we focused on DRM systems. In Chapter 6, we analysed the domain of DRM systems. There is no established consensus on security requirements for DRM systems, instead, the requirements were scattered throughout literature. From our analysis, we derived a set of core security requirements for DRM systems. First and foremost, they must provide security. This security must extend beyond a formal analysis – it must account for situations that deviate from the assumptions of the formal analysis. We found that one of the core requirements, fair exchange, was traditionally satisfied by the assumption of a client-server setting. Enabling fair exchange in non-client-server settings, such as a peer-to-peer setting, had not been studied in-depth.

In a peer-to-peer setting, the assurance of fair exchange enables content redistribution. Accordingly, the following research question was posed:

Research question II. *Is a formally and practically secure DRM system that enables content redistribution possible?*

This question was addressed in Chapter 7 by introducing Nuovo DRM, which enables content redistribution. Nuovo DRM was designed to satisfy the following goals:

- Effectiveness: the system functions correctly.
- Secrecy: no content is leaked.
- Resistance to content masquerading: it is impossible to pass one item of content off as another item.
- Strong fairness: No exchange of items between two parties can leave one party without its item and the other with two items.

These goals were formalised in regular μ -calculus, and the system was modelled in μ CRL. A finite instance of the system was verified via model-checking and found to satisfy the formalised goals.

The verification relies upon several assumptions, most notably the assumption of tamper-proof devices. As these assumptions cannot be guaranteed, we introduced several procedures that mitigate the effects of any breaking of the assumptions. The proposed mitigation procedures are the following:

- Resolution of client-to-client dispute.
- Detection of compromised devices.
- Isolation of compromised devices.

In conclusion, the goal of investigating the feasibility of a formally and practically secure DRM system was achieved by proposing such a system, formally verifying its security and proposing procedures to provide practical security.

8.2.1 Future work

There are several directions in which we wish to develop Nuovo DRM further. One of these is an actual implementation of Nuovo DRM. This has been successfully executed, but these results are outside the context of the work in this thesis.

Nuovo DRM, like other DRM systems, treats the content provider as a trusted source. An interesting extension is to investigate incorporating accountability of the provider.

Nuovo DRM, as described, abstracts away from rights semantics. Formalisation of rights expression languages is a heavily studied subject in DRM research. To the best of our knowledge, there has not yet been a formal verification of a DRM system that accounted in full detail for the formal rights language used. An extension of this is to incorporate the proposed revocation list mechanism in the formalisation.

Finally, the payment infrastructure is not detailed in Nuovo DRM. However, to comply with the mitigation procedures, such a payment scheme must satisfy several requirements. We have not investigated which payment schemes would satisfy the requirements, and what impact such a scheme has on the system.

Bibliography

- [AB03] M. Abadi and B. Blanchet. Computer-assisted verification of a protocol for certified email. In *Proc. 10th Symposium on Static Analysis, LNCS 2694*, pages 316–335. Springer, 2003.
- [AGGV05] G. Avoine, F. Gärtner, R. Guerraoui, and M. Vukolic. Gracefully degrading fair exchange with security modules. In *Proc. 5th European Dependable Computing Conference, LNCS 3463*, pages 55–71. Springer, 2005.
- [ALBD04] R. Aditya, B. Lee, C. Boyd, and E. Dawson. An efficient mixnet-based voting scheme providing receipt-freeness. In *Proc. 1st Conference on Trust and Privacy in Digital Business, LNCS 3184*, pages 152–161. Springer, 2004.
- [App06] A.W. Appel. How to defeat Rivest’s ThreeBallot voting system. Technical report, Princeton University, 2006.
- [AR06] B. Adida and R. L. Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In *Proc. 2006 ACM Workshop on Privacy in the Electronic Society*, pages 29–40. ACM, 2006.
- [ARS05] A. Adelsbach, M. Rohe, and A. Sadeghi. Towards multilateral secure digital rights distribution infrastructures. In *Proc. 5th ACM Digital Rights Management Workshop*, pages 45–54. ACM, 2005.
- [Aso98] N. Asokan. *Fairness in Electronic Commerce*. PhD thesis, University of Waterloo, Canada, 1998.
- [BCL⁺07] S. C. C. Blom, J. Calamé, B. Lissér, S. Orzan, J. Pang, J. C. van de Pol, M. Torabi Dashti, and A. J. Wijs. Distributed analysis with μ CRL, a compendium of case studies. In *Proc. 13th Conference on Tools and Algorithms for the Construction and Analysis of Systems, LNCS 4424*, pages 683–689. Springer, 2007.
- [BCT96] A. Basu, B. Charron-Bost, and S. Toueg. Simulating reliable links with unreliable links in the presence of process crashes. In *Proc. 10th ACM Workshop on Distributed Algorithms, LNCS 1151*, pages 105–122. Springer, 1996.

- [BFG⁺01] S. C. C. Blom, W. J. Fokkink, J. F. Groote, I. van Langevelde, B. Lisser, and J. C. van de Pol. μ CRL: A toolset for analysing algebraic specifications. In *Proc. 13th Conference on Computer Aided Verification, LNCS 2102*, pages 250–254. Springer, 2001.
- [BFP⁺01] O. Baudron, P. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *Proc. 20th ACM Symposium on Principles of Distributed Computing*, pages 274–283. ACM, 2001.
- [BHM08] M. Backes, C. Hrițcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proc. 21st Computer Security Foundations Symposium*, pages 195–209. IEEE Computer Society, 2008.
- [Bla79] G. R. Blakley. Safeguarding cryptographic keys. In *Proc. 1979 National Computer Conference*, vol. 48, pages 313–317. American Federation of Information Processing Societies, 1979.
- [BMR07] J. Bohli, J. Müller-Quade, and S. Röhrich. Bingo voting: Secure and coercion-free voting using a trusted random number generator. In *Proc. 1st Conference on Voting and Identity, LNCS 4896*, pages 111–124. Springer, 2007.
- [BP01] G. Bella and L. C. Paulson. Mechanical proofs about a non-repudiation protocol. In *Proc. 14th Conference on Theorem Proving in Higher Order Logics, LNCS 2152*, pages 91–104. Springer, 2001.
- [BRS07] A. Baskar, R. Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In *Proc. 11th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 62–71. ACM, 2007.
- [BT94] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th ACM Symposium on Theory of Computing*, pages 544–553. ACM, 1994.
- [BV96] J. W. Bakker and E. P. de Vink. *Control Flow Semantics*. Foundations of Computing Series. MIT Press, 1996.
- [BW90] J. C. M. Baeten and W. P. Weijland. *Process algebra, Cambridge Tracts in Theoretical Computer Science*, vol. 18. Cambridge University Press, 1990.
- [CA02] H. Chang and M. J. Atallah. Protecting software code by guards. In *Security and privacy in digital rights management, LNCS 2320*, pages 150–175. Springer, 2002.
- [CBM01] I. Cox, J. Bloom, and M. Miller. *Digital Watermarking: Principles & Practice*. Morgan Kaufmann, 2001.
- [CC97] L. F. Cranor and R. Cytron. Sensus: A security-conscious electronic polling system for the internet. In *Hawaii International Conference on System Sciences*, vol. 3, pages 561–570. IEEE Computer Society, 1997.

- [CCC⁺08] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, and A. T. Sherman. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *Preproceedings of the 2008 USENIX Electronic Voting Technology Workshop*, 2008.
- [CCM08] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *Proc. 29th Symposium on Security and Privacy*, pages 354–368. IEEE Computer Society, 2008.
- [CD06] D. J. T. Chong and R. H. Deng. Privacy-enhanced superdistribution of layered content with trusted access control. In *Proc. 6th ACM Digital Rights Management Workshop*, pages 37–44. ACM, 2006.
- [CEC⁺08] D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. *IEEE Security and Privacy*, 6(3):40–46, 2008.
- [Cer03] I. Cervesato. Data access specification and the most powerful symbolic attacker in MSR. In *Proc. 2002 Symposium on Software Security, LNCS 2609*, pages 384–416. Springer, 2003.
- [CFS⁺06] B. Chevallier-Mames, P. Fouque, J. Stern, D. Pointcheval, and J. Traoré. On some incompatible properties of voting schemes. In *Preproceedings WOTE*, 2006.
- [CFSY96] R. Cramer, M. K. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In *Advances in Cryptology – EUROCRYPT’96, LNCS 1070*, pages 72–83. Springer, 1996.
- [CGP00] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.
- [CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology – EUROCRYPT’97, LNCS 1233*, pages 103–118. Springer, 1997.
- [Cha81] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
- [Cha82] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of CRYPTO’82*, pages 199–203. Plenum, New York, 1982.
- [Cha04] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.
- [COPT07] T. Chothia, S.M. Orzan, J. Pang, and M. Torabi Dashti. A framework for automatically checking anonymity with μ CRL. In *Proc. 2nd Symposium on Trustworthy Global Computing, LNCS 4661*, pages 301–318. Springer, 2007.
- [Cre06] C. J. F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. PhD thesis, Eindhoven University of Technology, 2006.

- [CRS05] D. Chaum, P. Y. A. Ryan, and S. A. Schneider. A practical voter-verifiable election scheme. In *Proc. 10th European Symposium on Research in Computer Security, LNCS 3679*, pages 118–139. Springer, 2005.
- [CS02] H. Comon and V. Shmatikov. Is it possible to decide whether a cryptographic protocol is secure or not? *Journal of Telecommunications and Information Technology*, 4:3–13, 2002.
- [CSP⁺06] C. C. Chu, X. Su, B. S. Prabhu, R. Gadh, S. Kurup, G. Sridhar, and V. Sridhar. Mobile DRM for Multimedia Content Commerce in P2P Networks. In *Proc. 2nd CCNC Workshop on Digital Rights Management Impact on Consumer Communications*. IEEE Communications Society, 2006.
- [CT06] J. Cederquist and M. Torabi Dashti. An intruder model for verifying liveness in security protocols. In *Proc. 4th ACM Workshop on Formal Methods in Security Engineering*, pages 23–32. ACM, 2006.
- [DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
- [DGY03] N. Daswani, H. Garcia-Molina, and B. Yang. Open problems in data-sharing peer-to-peer systems. In *Proc. 9th Conference on Database Theory, LNCS 2572*, pages 1–15. Springer, 2003.
- [Din01] G. Dini. Electronic voting in a large-scale distributed system. *Networks*, 38(1):22–32, 2001.
- [DKR06] S. Delaune, S. Kremer, and M. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proc. 19th Computer Security Foundation Workshop*, pages 28–42. IEEE Computer Society, 2006.
- [DKR08] S. Delaune, S. Kremer, and M. Ryan. Verifying privacy-type properties of electronic voting protocols. Technical Report LSV-08-01, ENS Cachan, 2008.
- [DRS08] S. Delaune, M. Ryan, and B. Smyth. Automatic verification of privacy properties in the applied pi-calculus. In *Proc. 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security, IFIP Conference Proceedings*, vol. 263, pages 263–278. Springer, 2008.
- [DY83] D. Dolev and A. Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
- [ECA08] A. Essex, J. Clark, and C. Adams. Aperio: High integrity elections for developing countries. In *Preproceedings WOTE*, 2008. IAVoSS Workshop On Trustworthy Elections 2008, Leuven, Belgium.
- [Elg85] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proc. CRYPTO 84 on Advances in Cryptology, LNCS 218*, pages 10–18. Springer, 1985.

- [ES05] N. Evans and S. Schneider. Verifying security protocols with PVS: widening the rank function approach. *Journal of Logic and Algebraic Programming*, 64(2):253–284, 2005.
- [EY80] S. Even and Y. Yacobi. Relations among public key signature systems. Technical Report 175, Technicon, 1980.
- [FCS06] K. Fisher, R. Carback, and A. T. Sherman. Punchscan: Introduction and system definition of a high-integrity election system. In *Preproceedings WOTE*, 2006.
- [FG07] C. Fontaine and F. Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, 2007:1–10, 2007. Article ID 13801.
- [FGK⁺96] J.-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. CADP: A protocol validation and verification toolbox. In *Proc. 8th Conference on Computer Aided Verification, LNCS 1102*, pages 437–440. Springer, 1996.
- [FLP85] M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.
- [FMM⁺02] J. Furukawa, H. Miyauchi, K. Mori, S. Obana, and K. Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In *Proc. 12th Conference on Financial Cryptography and Data Security, LNCS 2357*, pages 16–30. Springer, 2002.
- [FOO92] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology – AUSCRYPT ’92, LNCS 718*, pages 244–251. Springer, 1992.
- [FS86] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO’86, LNCS 263*, pages 186–194. Springer, 1986.
- [FSS05] M. Fahrmaier, W. Sitou, and B. Spanfelner. Security and privacy rights management for mobile and ubiquitous computing. In *Workshop on Ubi-Comp Privacy*, pages 97–08, 2005.
- [GHH⁺08] B. Gedrojc, M. Hueck, H. Hoogstraten, M. Koek, and S. Resink. Advisering toelaatbaarheid internetstemvoorziening waterschappen. Technical report, Fox-IT BV, 2008.
- [GHPR05] F.D. Garcia, I. Hasuo, W. Pieters, and P. van Rossum. Provable anonymity. In *Proc. 3rd ACM Workshop on Formal Methods in Security Engineering*, pages 63–72. ACM, 2005.
- [GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
- [GN02] R. Grimm and J. Nützel. A friendly peer-to-peer file sharing system with profit but without copy protection. In *Proc. 2nd Workshop on Innovative Internet Computing Systems, LNCS 2346*, pages 133–142. Springer, 2002.

- [GP95] J. F. Groote and A. Ponse. The syntax and semantics of μ CRL. In *Algebra of Communicating Processes '94*, Workshops in Computing Series, pages 26–62. Springer, 1995.
- [GRV03] S. Gürgens, C. Rudolph, and H. Vogt. On the security of fair non-repudiation protocols. In *ISC '03, LNCS 2851*, pages 193–207. Springer, 2003.
- [Gut03] S. Guth. A sample DRM system. In *Digital Rights Management: Technical, Economical, Legal and Political Aspects, LNCS 2770*, pages 150–161. Springer, 2003.
- [Hea07] J. Heather. Implementing stv securely in pret a voter. In *CSF*, pages 157–169, 2007.
- [Her97] M. A. Herschberg. Secure Electronic Voting over the World Wide Web. Master's thesis, Massachusetts Institute of Technology, 1997.
- [Hir01] M. Hirt. *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*. PhD thesis, ETH Zurich, 2001. Reprinted as vol. 3 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-747-2, Hartung-Gorre, 2001.
- [HJ08] G. L. Heileman and M. Joye, editors. *Proc. of the 8th ACM Workshop on Digital Rights Management 2008*. ACM, 2008.
- [HJP05] E. Hubbers, B. Jacobs, and W. Pieters. RIES - internet voting in action. In *Proc. 29th Computer Software and Applications Conference*, vol. 1, pages 417–424. IEEE Computer Society, 2005.
- [HJS⁺08] E Hubbers, B Jacobs, B Schoenmakers, H. Van Tilborg, and B. De Weger. Description and analysis of the RIES internet voting system. Technical report, Eindhoven Institute for the Protection of Systems and Information (EiPSI), 2008.
- [HK02] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system. In *Proc. 3rd Conference on Music Information Retrieval*. IRCAM, 2002.
- [HLS00] J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *Proc. 13th Computer Security Foundations Workshop*, pages 255–268. IEEE Computer Society, 2000.
- [HMST02] B. Horne, L. Matheson, C. Sheehan, and R. E. Tarjan. Dynamic self-checking techniques for improved tamper resistance. In *Security and privacy in digital rights management, LNCS 2320*, pages 141–159. Springer, 2002.
- [HS00] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptology – EUROCRYPT 2000, LNCS 1807*, pages 539–556. Springer, 2000.

- [IAUS03] T. Iwata, T. Abe, K. Ueda, and H. Sunaga. A DRM system suitable for P2P content delivery and the study on its implementation. In *Proc. 9th Asia-Pacific Conference on Communications*, vol. 2, pages 806–811. IEEE, 2003.
- [JCJ05] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Proc. 2005 ACM Workshop on Privacy in the Electronic Society*, pages 61–70. ACM, 2005.
- [JJR02] M. Jakobsson, A. Juels, and R. L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proc. 11th USENIX Security Symposium*, pages 339–353. USENIX, 2002.
- [JKT06] H. L. Jonker, S. Krishnan Nair, and M. Torabi Dashti. Nuovo DRM paradiso. Technical Report SEN-R0602, Centrum voor Wiskunde en Informatica, Netherlands, 2006.
- [JM07] H. L. Jonker and S. Mauw. Core security requirements of DRM systems. In *Digital Rights Management – An Introduction*, pages 73–90. ICFAI University Press, 2007.
- [JM09] H. L. Jonker and S. Mauw. Discovering the core security requirements of DRM systems by means of objective trees. In *Handbook of Research on Secure Multimedia Distribution*, pages 71–85. IGI Global, 2009.
- [JMP09a] H. L. Jonker, S. Mauw, and J. Pang. A formal framework for quantifying voter-controlled privacy. *Journal of Algorithms in Cognition, Informatics and Logic*, 64(2-3):89–105, 2009.
- [JMP09b] H. L. Jonker, S. Mauw, and J. Pang. Measuring voter-controlled privacy. In *Proc. Conference on Availability, Reliability and Security 2009*, pages 289–298. IEEE Computer Society, 2009.
- [Jon04] H. L. Jonker. Security of Digital Rights Management systems. Master’s thesis, Eindhoven University of Technology, 2004.
- [JP06] H. L. Jonker and W. Pieters. Receipt-freeness as a special case of anonymity in epistemic logic. In *Preproceedings WOTE*, 2006.
- [JV06] H. L. Jonker and E. P. de Vink. Formalising Receipt-Freeness. In *Proc. 9th Information Security Conference, LNCS 4176*, pages 476–488. Springer, 2006.
- [JV07] H. L. Jonker and M. Volkamer. Compliance of RIES to the proposed e-voting protection profile. In A. Alkassar and M. Volkamer, editors, *Proc. 1st Conference on Voting and Identity, LNCS 4896*, pages 50–61. Springer, 2007. Bochum, Germany.
- [KEH⁺04] T. Kalker, D. H. J. Epema, P. H. Hartel, R. L. Lagendijk, and M. van Steen. Music2Share - Copyright-compliant music sharing in P2P systems. *Proceedings of the IEEE*, 92(6):961–970, 2004.

- [KK05] D. Kähler and R. Küsters. Constraint solving for contract-signing protocols. In *Proc. 16th Conference on Concurrency Theory, LNCS 3653*, pages 233–247. Springer, 2005.
- [KMZ02] S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of non-repudiation protocols. *Computer Communications*, 25(17):1606–1621, 2002.
- [KR01] S. Kremer and J. Raskin. A game-based verification of non-repudiation and fair exchange protocols. In *Proc. 13th Conference on Concurrency Theory, LNCS 2154*, pages 551–565. Springer, 2001.
- [KR05] S. Kremer and M. Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *Proc. 14th European Symposium on Programming, LNCS 3444*, pages 186–200. Springer, 2005.
- [KW99] J. Karro and J. Wang. Towards a practical, secure, and very large scale online election. In *Proc. 15th Annual Computer Security Applications Conference*, pages 161–169. IEEE Computer Society, 1999.
- [KY02] A. Kiayias and M. Yung. Self-tallying elections and perfect ballot secrecy. In *Proc. 5th Workshop on Practice and Theory in Public Key Cryptosystems, LNCS 2274*, pages 141–158. Springer, 2002.
- [KY03] A. Kiayias and M. Yung. Breaking and repairing asymmetric public-key traitor tracing. In *Digital Rights Management, LNCS 2320*, pages 32–50. Springer, 2003.
- [Lam77] L. Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, 3(2):125–143, 1977.
- [Lan77] G. Lann. Distributed systems - towards a formal approach. In *Information Processing 77, Proceedings of the IFIP Congress*, pages 155–160. Elsevier North-Holland, 1977.
- [LBD⁺03] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Proc. 5th Conference on Information and Communications Security, LNCS 2836*, pages 245–258. Springer, 2003.
- [LGK02] C. Lambrinoudakis, D. Gritzalis, and S. K. Katsikas. Building a reliable e-voting system: Functional requirements and legal constraints. In *Proc. 13th Workshop on Database and Expert Systems Applications Workshops*, page 435. IEEE Computer Society, 2002.
- [LK00] B. Lee and K. Kim. Receipt-free electronic voting through collaboration of voter and honest verifier. In *Proc. 2000 Joint workshop on Information Security and Cryptology*, pages 101–108. Institute of Electronics, Information and Communication Engineers Japan, 2000.
- [LK02] B. Lee and K. Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In *Proc. 2002 Information and Communications Security Conference, LNCS 2587*, pages 389–406. Springer, 2002.

- [LKK⁺03] C. Lambrinoudakis, S. Kokolakis, M. Karyda, V. Tsoumas, D. Gritzalis, and S. K. Katsikas. Electronic voting systems: Security implications of the administrative workflow. In *Proc. 14th Workshop on Database and Expert Systems Applications Workshops, LNCS 2763*, pages 467–471. Springer, 2003.
- [LR08a] D. Lundin and P. Y. A. Ryan. Human readable paper verification of prêt à voter. In *Proc. 13th European Symposium on Research in Computer Security, LNCS 5283*, pages 379–395. Springer, 2008.
- [LR08b] D. Lundin and P. Y. A. Ryan. Human readable paper verification of prêt à voter. In *ESORICS*, pages 379–395, 2008.
- [MBC01] E. Magkos, M. Burmester, and V. Chrissikopoulos. Receipt-freeness in large-scale elections without untappable channels. In *IFIP Conference on E-Commerce/E-business/E-Government (I3E)*, pages 683–694. Kluwer, 2001.
- [Mea03] C. Meadows. Formal methods for cryptographic protocol analysis: Emerging issues and trends. *IEEE Journal of Selected Areas in Communication*, 21(1):44–54, 2003.
- [MH96] M. Michels and P. Horster. Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In *Advances in Cryptology - ASIACRYPT '96, LNCS 1163*, pages 125–132. Springer, 1996.
- [MS03] R. Mateescu and M. Sighireanu. Efficient on-the-fly model-checking for regular alternation-free μ -calculus. *Science of Computer Programming*, 46(3):255–281, 2003.
- [MVV04] S. Mauw, J. Verschuren, and E. P. de Vink. A formalization of anonymity and onion routing. In *Proc. 9th European Symposium on Research Computer Security, LNCS 3193*, pages 109–124. Springer, 2004.
- [Nef01] C. A. Neff. A verifiable secret shuffle and its application to e-voting. In *Proc. 8th Conference on Computer and Communications Security*, pages 116–125. ACM, 2001.
- [NPG⁺05] S. Nair, B. Popescu, C. Gamage, B. Crispo, and A. Tanenbaum. Enabling DRM-preserving digital content redistribution. In *Proc. 7th Conference on E-Commerce Technology*, pages 151–158. IEEE Computer Society, 2005.
- [NR94] V. Niemi and A. Renvall. How to prevent buying of votes in computer elections. In *Advances in Cryptology - ASIACRYPT'94, LNCS 917*, pages 164–170. Springer, 1994.
- [NZJK06] S. Nepal, J. J. Zic, F. Jaccard, and G. Kraehenbuehl. A tag-based data model for privacy-preserving medical applications. In *1st EDBTW Workshop on Information in Healthcare*, pages 433–444, 2006.
- [Oka96] T. Okamoto. An electronic voting scheme. In *Proc. IFIP World Conference on IT Tools*, pages 21–30. Chapman & Hall, 1996.

- [Oka97] T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. 1997 Security Protocols Workshop, LNCS 1361*, pages 25–35. Springer, 1997.
- [OMA04] The Open Mobile Alliance OMA. OMA-DRM-ARCH-V2.0-20040715-C – DRM architecture, 2004.
- [Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT’99, LNCS 1592*, pages 223–238. Springer, 1999.
- [PKCT04] B. C. Popescu, F. L. A. J. Kamperman, B. Crispo, and A. S. Tanenbaum. A DRM security architecture for home networks. In *Proc. 4th ACM Workshop on Digital Rights Management*, pages 1–10. ACM, 2004.
- [Plo81] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University Denmark, 1981.
- [Plo04] G. D. Plotkin. The origins of structural operational semantics. *J. Log. Algebr. Program.*, 60-61:3–15, 2004.
- [Pou99] A. Pouloudi. Aspects of the stakeholder concept and their implications for information systems development. In *Proc. 32nd Hawaii International Conference on System Sciences*. IEEE Computer Society, 1999.
- [Pri90] R. Prieto-Díaz. Domain analysis: An introduction. *Software Engineering Notes*, 15(2):47–54, 1990.
- [PT01] L. Prechelt and R. Typke. An interface for melody input. *ACM Transactions on Computer-Human Interaction*, 8(2):133–149, 2001.
- [PV04] J. C. van de Pol and M. Valero Espada. Modal abstractions in μ CRL. In *Proc. 10th Conference on Algebraic Methodology and Software Technology, LNCS 3116*, pages 409–425. Springer, 2004.
- [PVG03] H. Pagnia, H. Vogt, and F. Gärtner. Fair exchange. *Computer Journal*, 46(1):55–75, 2003.
- [PXZ02] A. Pnueli, J. Xu, and L. D. Zuck. Liveness with $(0, 1, \infty)$ -counter abstraction. In *Proc. 14th International Conference on Computer Aided Verification, LNCS 2304*, pages 107–122. Springer, 2002.
- [RS04a] T. Reti and R. Sarvas. *DiMaS*: distributing multimedia on peer-to-peer file sharing networks. In *Proc. 12th Annual ACM Conference on Multimedia*, pages 166–167. ACM, 2004.
- [RS04b] T. Reti and R. Sarvas. The *dimas* system for authoring communities: distributing semantic multimedia content on peer-to-peer file sharing networks. In *Web Intelligence—Proceedings of the 11th Finnish AI Conference*. Finnish AI Society, 2004.
- [RS06] P. Y. A. Ryan and S. A. Schneider. Prêt à voter with re-encryption mixes. In *Proc. 11th European Symposium on Research in Computer Security, LNCS 4189*, pages 313–326. Springer, 2006.

- [RS07] R. L. Rivest and W. D. Smith. Three voting protocols: ThreeBallot, VAV, and Twin. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop*. USENIX, 2007.
- [RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [Rya04] P.Y.A. Ryan. A variant of the chaum voting scheme. Technical Report CS-TR-864, University of Newcastle upon Tyne, 2004.
- [Rya05] P. Y. A. Ryan. A variant of the chaum voter-verifiable scheme. In *Proc. 2005 Workshop on Issues in the Theory of Security*, pages 81–88, 2005.
- [Rya08] P.Y.A. Ryan. Prêt à voter with paillier encryption. *Mathematical and Computer Modelling*, 48(9-10):1646–1662, 2008. Mathematical Modeling of Voting Systems and Elections: Theory and Applications.
- [Sch00] B. Schneier. *Secrets & Lies: Digital Security in a Networked World*. Wiley, 2000.
- [SGR97] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *Proc. 1997 Symposium on Security and Privacy*, pages 44–54. IEEE Computer Society, 1997.
- [Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
- [SK95] K. Sako and J. Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In *Advances in Cryptology – EUROCRYPT’95, LNCS 921*, pages 393–403. Springer, 1995.
- [SM02] V. Shmatikov and J. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science*, 283(2):419–450, 2002.
- [SNB⁺03] C. Serrão, D. Naves, T. Barker, M. Balestri, and P. Kudumakis. Open SDRM – an open and secure digital rights management solution. In *Proc. 2003 IADIS Conference on E-society*. IADIS Press, 2003.
- [Som04] I. Sommerville. *Software Engineering*. Pearson, 2004.
- [SV02] W. Shapiro and R. Vingralek. How to manage persistent state in DRM systems. In *Security and privacy in digital rights management, LNCS 2320*, pages 176–191. Springer, 2002.
- [SW02] R. Safavi-Naini and Y. Wang. Traitor tracing for shortened and corrupted fingerprints. In *Proc. ACM 2001 Digital Rights Management Workshop, LNCS 2320*, pages 32–50. Springer, 2002.
- [TKJ07] M. Torabi Dashti, S. Krishnan Nair, and H. L. Jonker. Nuovo DRM paradiso: Towards a verified fair DRM scheme. In *Proc. 1st Symposium on Fundamentals of Software Engineering, LNCS 4767*, pages 33–48. Springer, 2007.

- [TKJ08] M. Torabi Dashti, S. Krishnan Nair, and H. L. Jonker. Nuovo DRM paradiso: Designing a secure, verified, fair exchange DRM scheme. *Fundamenta Informaticae*, 89(4):393–417, 2008.
- [Wik03] D. Wikström. Five practical attacks for “optimistic mixing for exit-polls”. In *Proc. 10th Workshop on Selected Areas in Cryptography, LNCS 3006*, pages 160–175. Springer, 2003.
- [XSH⁺07] Z. Xia, S. A. Schneider, J. Heather, P. Y. A. Ryan, D. Lundin, R. Peel, and P. Howard. Prêt à Voter: All-In-One. *Proceedings of IAVoSS Workshop on Trustworthy Elections (WOTE 2007)*, pages 47–56, 2007.
- [YKS06] M. Yung, K. Kurosawa, and R. Safavi-Naini, editors. *Proc. of the 6th ACM Workshop on Digital Rights Management Workshop*. ACM, 2006.
- [YKS07] M. Yung, A. Kiayias, and A. Sadeghi, editors. *Proc. of the 7th ACM Workshop on Digital Rights Management*. ACM, 2007.

Index of subjects

- μ CRL, 57, 122, 131
 - syntax, 131
- abstract actions, 130
- ACP, 57
- agent, 57
 - semantics, 60
 - state, 59
- anonymous receive, 61
- anonymous send, 61
- applied π , 52
- authorities, 10
- ballot, 10
- blind signatures, 19, 73
- cast, 10
- choice function, 54, 63
- choice group, 67
- choice indistinguishability, 67
- coercion-resistance, 52
- compliant device, 123
- conditional choice, 57, 62
- content, 105
- Counter, 10, 84
- deadlock, 57
- derivability, 56
- designated verifier, *see* proofs
- device revocation list, 144
- Digital Rights Management, *see* DRM
- Dolev-Yao, 54
- DRL, *see* device revocation list
- DRM, 105
 - concept, 108
- DVP, *see* proofs
- election, 10
- epistemic logic, 51
- events, 57
 - phase, 57
- receive, 57
- send, 57
- fairness, 126, 137
- forced abstention, 52
- framework
 - extensions, 72
- function fv, 57
- fv function, 58
- guarded recursion, 57, 62
- homomorphic encryption, 18, 73
- incentives
 - content creator, 108
 - distributor, 109
 - user, 109
- initial state, 63
- intruder model, 54, 134
- knowledge, 56
- knowledge sharing, 67
- license, 112
- matching, 55
- mix, 24
- mixnet, 24
- naming convention, 58
- non-deterministic choice, 57, 62
- option, candidate, 10
- phase synchronisation, 62
- process model, 112
 - basic, 112
 - generic, 113
 - refined, 112
- process variables, 57
- processes, 57
- projection function, 54, 59
- proofs, 21, 75

- designated verifier, 22, 75
 - zero knowledge, 21, 75
- public receive, 61
- public send, 61
- randomisation attack, 52
- re-encryption, 20, 74
- readability, 60
- receipt, 52, 67
- receipt-freeness, 52, 67
- Registrar, 10, 84, 94
- reinterpretation, 66
- requirements
 - accuracy, 14
 - coercion-resistance, 15
 - democracy, 14
 - eligibility, 14
 - forced abstention, 15
 - individual verifiability, 14
 - of content creator, 114
 - of distributor, 114
 - of user, 116
 - random voting, 15
 - receipt-freeness, 15
 - simulation, 15
 - universal verifiability, 14
 - vote privacy, 14
- secret sharing, 22
 - shared decryption, 23, 74
- security properties, 109
 - content creator, 110
 - distributor, 110
 - overview, 111
 - user, 110
- semantics, 60
- simulation attack, 52
- stakeholders, 107
- substitution, 55, 58
- system semantics, 63
- tally, 10
- terms, 55
- traces, 64
 - indistinguishability, 67
 - visible part of, 66
- untappable receive, 62
- untappable send, 61
- verifiability, 97
- vote, 10
- voter, 10
 - conspiring, 67
- voting credentials, 10
- voting schemes
 - CGS97, 27
 - FOO92, 27, 38, 39
 - description, 81
 - Evex, 40
 - model, 82
 - privacy analysis, 85
 - Sensus, 40
 - receipt-free, 27
 - ALBD04, 37–39
 - BFP01, 34, 38, 39
 - BT94, 29, 38, 39
 - CGS97, 38, 39
 - HS00, 32, 38, 39, 97
 - JCJ05, 37–39
 - KY02, 35, 38, 39
 - LBD03, 36, 38, 39
 - LK00, 33, 38, 39
 - LK02, 36, 38, 39
 - MBC01, 34, 38, 39
 - NR94, 30, 38, 39
 - Oka97, 31, 38, 39
 - SK95, 30, 38, 39
- voting system, 59
 - state, 63
- voting systems, 26
 - end-to-end, 40
 - Prêt à Voter, 89
 - 3BS, 47
 - Aperio, 48
 - Bingo Voting, 48
 - Chaum04, 42
 - DigiShuff, 41
 - Prêt à Voter, 42
 - Punchscan, 44
 - Scantegrity, 45
 - Scratch & Vote, 46
 - RIES, 41
- ZKP, *see* proofs

Summary

Security Matters: Privacy in Voting and Fairness in Digital Exchange

Security matters. In the real world, there are various basic mechanisms that provide security. The digital counterparts of these basic mechanisms have been studied and are now well understood. Understanding of more complex security notions has yet to mature. This thesis studies two complex security notions: privacy in voting and fairness in digital exchange. To facilitate the two different domains, the thesis discusses each notion in its own part.

Part I: Privacy in Voting

Voting systems have been evolving for years, incorporating new procedures and new countermeasures to detected problems. When the field of voting emerged in the 80s as a research area in computer science, it was not straightforward to transform these procedures and countermeasures into clearly specified, exact requirements necessary in computer science. Thus, requirements for voting systems had to be rediscovered in computer science over the years.

Without a high-level overview of the field, it remains unclear whether the set of requirements covers all desired properties. To this end, we analysed the domain of voting (cf. Chapter 2). The analysis resulted in a set of high-level security requirements for voting systems. Amongst these is the requirement *All voters are free to cast their votes*. Privacy is an essential ingredient to ensure this: if no one can determine how a voter voted, no one can hold that voter accountable for her vote. Ensuring privacy thus prevents any restrictions on freedom targeted to specific voters. Unfortunately, the concept of privacy is not well understood in the field of voting. There are various terms which all cover an aspect of privacy. We set out to investigate and improve the understanding of privacy in voting.

To investigate current understanding of privacy in voting, we surveyed a number of voting system proposals from literature (cf. Chapter 3). We found that while every system claims some form of privacy, several of these claims were dubious and, in several cases, these claims were broken by later works. In addition, the way privacy is specified varies from system to system. There is no unique notion of privacy in voting. This lack of consensus became more apparent when the risk of vote selling was introduced. In vote selling, privacy prevents a voter from proving how she voted, which means that selling her vote is impossible. Thus, a vote seller will actively try to reduce her privacy. Voting systems should prevent voters from doing so. However, failing a consensus on privacy, it is unclear how to expand the notion of privacy to incorporate vote selling. Thus, it is unclear how to verify that systems enforce sufficient privacy to prevent vote selling. This necessitates a new understanding of privacy.

To this end, we introduce a formal framework that determines precisely for whom a voter could have voted (cf. Chapter 4). By comparing a regular voter with a voter who tries to reduce her privacy, we can determine if a voter can reduce her privacy and if so, by how much. In this fashion, the privacy of a voter was quantified. Several existing notions of privacy were easily captured in this way. But the preciseness of the framework extends beyond that: it also can detect partial loss of privacy.

In order to show how the framework and the ideas it embodies improve understanding of voting, we apply the framework and its concepts in Chapter 5. The framework is used to ascertain the privacy of a purely theoretical scheme (FOO92), and of an actually implemented voting system (Prêt à Voter), and the concept of choice groups is used to discuss the relation between privacy and verifiability in the 3BS voting system. Using the framework, we prove that FOO92 offers privacy to voters, but it does not enforce it: vote selling is possible. The framework also clarifies the reasons for this, and based on this indication, we sketch a possible approach for improving privacy of FOO92. The Prêt à Voter voting system uses paper ballots and a voting booth. With the help of the framework, the impact of these physical measures on privacy is made explicit. Thus, we can reason about adapting Prêt à Voter for remote voting, where such measures are not necessarily possible. Finally, the 3BS system uses a novel way to provide a trade off between privacy and verifiability. Using the concept of choice groups, we gain a better understanding of how much privacy is sacrificed.

Part II: Fairness in Digital Exchange

There exist various forms of controlling access to items. This is for example necessary in trade – without scarcity of items, there is no need to trade for them. With the advent of the Internet, a large-scale distributed system, the question of how to enforce access control to digital items in a distributed environment emerged, as a way to enable online trade. One answer was to distribute access control together with the controlled items. This is what Digital Rights Management (DRM) sets out to achieve. As such, DRM is to enable online trade in digital items.

Again, a high-level view of the field is necessary to obtain some certainty that the set of requirements for DRM systems covers all desired aspects. In Chapter 6, the domain of DRM is analysed, resulting in a set of security requirements. We highlight two requirements: the *impact of breaks is constrained* and *fair exchange*. The fair exchange of two items a and b between two parties ensures that either a and b both switch owners, or neither does. In the usual client-server (client-seller) setting of DRM systems, fair exchange is only ensured for the seller: the seller receives the (assurance of) payment and then proceeds to deliver. Recently, the idea of DRM has emerged in other settings, such as a peer-to-peer (client-to-client) setting. In such a setting, guaranteeing fair exchange for both seller and buyer becomes an important prerequisite for trading.

In Chapter 7, we develop Nuovo DRM. Nuovo DRM is a secure DRM system that enables fair exchange and constrains the impact of breaks. As claims of security must be substantiated, we do so using formal methodology. We formalise the goals of Nuovo DRM and verify (using model checking) that these goals are achieved by an abstract model of Nuovo DRM. In addition to that, we present several procedures that mitigate the effects of breaks.

Samenvatting

Security Matters: Privacy in Voting and Fairness in Digital Exchange

Veiligheid is belangrijk. In de echte wereld zijn er verschillende basismechanismes die voor veiligheid zorgen. De digitale evenknieën van deze basismechanismes zijn in de informatica uitvoerig bestudeerd en worden tegenwoordig goed begrepen. Complexere veiligheidsbegrippen worden daarentegen nog niet goed begrepen. Dit proefschrift bestudeert twee complexe security begrippen: anonimiteit in stemsystemen en eerlijkheid in digitale uitwisseling. Het proefschrift wijdt een apart deel aan ieder van deze twee domeinen.

Deel I: Anonimiteit in stemsystemen

Stemsystemen zijn in de loop der jaren geëvolueerd en omvatten inmiddels allerlei maatregelen en procedures tegen fraude. Toen het onderzoeksgebied van elektronisch stemmen in de jaren 80 ontstond in de informatica, was er geen eenvoudige vertaling van deze procedures en maatregelen naar eenduidig gespecificeerde, exacte eisen (zoals nodig in de informatica) voorhanden. Dit leidde tot het gestaag herontdekken van eisen aan stemsystemen in de informatica.

Echter, zonder een overzicht van het onderzoeksgebied blijft het onduidelijk of de herontdekte eisen alle gewenste aspecten afdekken. Dientengevolge analyseerden we het gebied van elektronisch stemmen in Hoofdstuk 2. De analyse resulteerde in een verzameling eisen voor stemsystemen. We richtten ons op de eis *alle stemmers zijn vrij hun stem uit te brengen*. Anonimiteit is een essentiële voorwaarde voor deze vrijheid: een stemmer kan niet aansprakelijk gehouden worden voor haar stem, als haar stem anoniem is. Het garanderen van anonimiteit voorkomt dus beperkingen van de stemvrijheid die gericht zijn op individuele stemmers. Helaas blijkt anonimiteit niet goed te worden begrepen. Zo zijn er verschillende termen, die allen bepaalde delen van anonimiteit wel dekken, en andere delen niet. We stellen het doel om het begrip van anonimiteit in stemmen te onderzoeken en te verbeteren.

Ten eerste onderzochten we in Hoofdstuk 3 het huidige begrip van anonimiteit, door middel van een literatuurstudie van stemsystemen. Hoewel ieder systeem beweert een vorm van anonimiteit te bieden, blijkt echter dat sommige van deze beweringen dubieus waren, en in een aantal gevallen zijn deze zelfs later weerlegd. Daarbovenop bleek dat de wijze waarop anonimiteit is gespecificeerd, varieert van systeem tot systeem. Er is geen uniek, eenduidig concept van anonimiteit in stemsystemen. Dit gebrek aan consensus werd nog duidelijker, toen het verkopen van stemmen werd herkend als een veiligheidsrisico. Als een stemmer anoniem is, kan ze niet aan de stemmenkoper bewijzen hoe ze heeft gestemd, en dus kan ze haar stem niet verkopen. Een stemmer die haar stem wil verkopen, zal dus proberen om haar anonimiteit te

verminderen. Stemsystemen dienen dit te verhinderen. Door het gebrek aan een overeenstemming over het begrip anonimiteit is het onduidelijk hoe men kan bepalen, of een systeem genoeg anonimiteit garandeert om stemmenverkoop te verhinderen. Er is dus een beter begrip van anonimiteit nodig.

Hiertoe introduceren we een formeel framework in Hoofdstuk 4, dat precies bepaalt op welke kandidaat een stemmer gestemd zou kunnen hebben. Door een reguliere stemmer te vergelijken met een stemmer die haar anonimiteit probeert te reduceren, kunnen we vaststellen of, en in hoe verre, een stemmer haar anonimiteit kan verminderen. Op deze wijze is de anonimiteit van een stemmer gekwantificeerd. Bestaande anonimiteitsbegrippen laten zich eenvoudig vertalen naar het framework. Maar de precisie van het framework streeft bestaande begrippen voorbij: ook partieel verlies van anonimiteit wordt gedetecteerd.

We tonen een aantal toepassingen van het framework en de onderliggende begrippen in Hoofdstuk 5, om te verduidelijken hoe deze zaken het begrip van anonimiteit verbeteren. Het framework wordt toegepast op een theoretisch stelsysteem, het FOO92 systeem, en op een bestaand systeem, het Prêt à Voter systeem. Het begrip keuze-groep wordt gebruikt om de complexe relatie tussen anonimiteit en verifieerbaarheid in het 3BS systeem te verhelderen. Met behulp van het framework wordt duidelijk dat FOO92 anonimiteit aanbiedt, maar niet afdwingt: verkopen van stemmen is mogelijk. Het framework maakt inzichtelijk waar FOO92 versterkt dient te worden. Aan de hand hiervan suggereren we een mogelijke verbetering van FOO92. Het Prêt à Voter systeem maakt gebruik van een stemhok en papieren stembiljetten. Dankzij het framework wordt de impact van deze bouwstenen op anonimiteit verduidelijkt. Hierdoor wordt het mogelijk om te redeneren over het aanpassen van Prêt à Voter voor thuisstemmen, waar dit soort fysieke maatregelen niet mogelijk zijn.

Deel II: Eerlijkheid in digitale uitwisseling

Er zijn verschillende manieren waarop gecontroleerde toegang tot objecten wordt bewerkstelligd. Gecontroleerde toegang is bijvoorbeeld noodzakelijk voor handel – als iedereen bij alle objecten kan, is er geen reden tot handel. Met de opkomst van het Internet, een grootschalig gedistribueerd systeem, rees de vraag hoe toegangscontrole van digitale objecten gewaarborgd kan worden om online handel mogelijk te maken. Eén antwoord was om toegangscontrole samen te voegen met de digitale objecten. Dit is het doel van Digital Rights Management (DRM).

Ook voor dit onderzoeksdomein is een overzicht nodig om te verzekeren dat de verzameling eisen aan DRM systemen alle gewenste aspecten omvat. In Hoofdstuk 6 wordt het DRM-domein geanalyseerd. Dit levert een verzameling eisen op, waarvan we ons hoofdzakelijk richten op de eisen *de impact van succesvolle aanvallen wordt beperkt* en *eerlijk oversteken*. Eerlijk oversteken van twee objecten a en b tussen twee partijen garandeert dat of a en b beide van eigenaar wisselen, of dat geen van de twee van eigenaar wisselt. Eerlijk oversteken voorkomt dus de situatie waarin na afloop één partij zowel a als b bezit. In de gebruikelijke client-server (klant-verkoper) context van DRM systemen is eerlijk oversteken enkel gegarandeerd voor de verkoper: pas na ontvangst van (garantie van) betaling wordt geleverd. Recentelijk is een andere context, peer-to-peer (klant-klant) verkend voor DRM. In een dergelijke context is de garantie van eerlijk oversteken een belangrijke voorwaarde voor handel.

In Hoofdstuk 7 ontwikkelen we Nuovo DRM. Nuovo DRM is een veilig DRM systeem dat eerlijk oversteken mogelijk maakt en de impact van succesvolle aanvallen beperkt. Aangezien beweringen van veiligheid onderbouwd dienen te worden, doen we dit gebruikmakend van formele methoden. De doelen van Nuovo DRM worden geformaliseerd en met behulp van model checking verifiëren we dat een abstract model van Nuovo DRM deze doelen bewerkstelligt. Daarnaast presenteren we verschillende procedures die de impact van hacks verminderen.

Curriculum Vitae

- 2009 Research Fellow in the Formal Methods and Security group of Prof. Dr. S. Schneider, Faculty of Engineering and Physical Sciences, University of Surrey, United Kingdom.
- 2007–2009 Ph.D. student in the Security and Trust of Software Systems Group of Prof. Dr. S. Mauw, Faculty of Science, Technology and Communication, University of Luxembourg, Luxembourg.
- 2004–2008 Ph.D. student in the Formal Methods group of Prof. Dr. J.C.M. Baeten, Department of Mathematics and Computer Science, Eindhoven University of Technology, the Netherlands.
- 1996–2004 M.Sc. degree in Computer Science, Eindhoven University of Technology, the Netherlands. Title of Master's thesis: *Security of Digital Rights Management Systems*. Supervisors: Dr. S. Mauw, ir. A.T.S.C. Schoonen, ir. J.H.S. Verschuren.
- 1989–1996 Secondary school: Lorentz Lyceum, Eindhoven, the Netherlands. Born on 9th of August, 1978 in Eindhoven, the Netherlands.

Titles in the IPA Dissertation Series since 2005

- E. Ábrahám.** *An Assertional Proof System for Multithreaded Java -Theory and Tool Support-*. Faculty of Mathematics and Natural Sciences, UL. 2005-01
- R. Ruimerman.** *Modeling and Remodeling in Bone Tissue.* Faculty of Biomedical Engineering, TU/e. 2005-02
- C.N. Chong.** *Experiments in Rights Control - Expression and Enforcement.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03
- H. Gao.** *Design and Verification of Lock-free Parallel Algorithms.* Faculty of Mathematics and Computing Sciences, RUG. 2005-04
- H.M.A. van Beek.** *Specification and Analysis of Internet Applications.* Faculty of Mathematics and Computer Science, TU/e. 2005-05
- M.T. Ionita.** *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures.* Faculty of Mathematics and Computing Sciences, TU/e. 2005-06
- G. Lenzini.** *Integration of Analysis Techniques in Security and Fault-Tolerance.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07
- I. Kurtev.** *Adaptability of Model Transformations.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08
- T. Wolle.** *Computational Aspects of Treewidth - Lower Bounds and Network Reliability.* Faculty of Science, UU. 2005-09
- O. Tveretina.** *Decision Procedures for Equality Logic with Uninterpreted Functions.* Faculty of Mathematics and Computer Science, TU/e. 2005-10
- A.M.L. Liekens.** *Evolution of Finite Populations in Dynamic Environments.* Faculty of Biomedical Engineering, TU/e. 2005-11
- J. Eggermont.** *Data Mining using Genetic Programming: Classification and Symbolic Regression.* Faculty of Mathematics and Natural Sciences, UL. 2005-12
- B.J. Heeren.** *Top Quality Type Error Messages.* Faculty of Science, UU. 2005-13
- G.F. Frehse.** *Compositional Verification of Hybrid Systems using Simulation Relations.* Faculty of Science, Mathematics and Computer Science, RU. 2005-14
- M.R. Mousavi.** *Structuring Structural Operational Semantics.* Faculty of Mathematics and Computer Science, TU/e. 2005-15
- A. Sokolova.** *Coalgebraic Analysis of Probabilistic Systems.* Faculty of Mathematics and Computer Science, TU/e. 2005-16
- T. Gelsema.** *Effective Models for the Structure of π -Calculus Processes with Replication.* Faculty of Mathematics and Natural Sciences, UL. 2005-17
- P. Zoetewij.** *Composing Constraint Solvers.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-18
- J.J. Vinju.** *Analysis and Transformation of Source Code by Parsing and Rewriting.* Faculty of Natural Sciences,

Mathematics, and Computer Science, UvA. 2005-19

M. Valero Espada. *Modal Abstraction and Replication of Processes with Data.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell.* Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22

E. Dolstra. *The Purely Functional Software Deployment Model.* Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems.* Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems.* Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality.* Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications.* Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting.* Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs.* Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations.* Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data.* Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space.* Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices.* Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization.* Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML.* Faculty of Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

- B. Gebremichael.** *Expressivity of Timed Automata Models.* Faculty of Science, Mathematics and Computer Science, RU. 2006-18
- L.C.M. van Gool.** *Formalising Interface Specifications.* Faculty of Mathematics and Computer Science, TU/e. 2006-19
- C.J.F. Cremers.** *Scyther - Semantics and Verification of Security Protocols.* Faculty of Mathematics and Computer Science, TU/e. 2006-20
- J.V. Guillen Scholten.** *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition.* Faculty of Mathematics and Natural Sciences, UL. 2006-21
- H.A. de Jong.** *Flexible Heterogeneous Software Systems.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01
- N.K. Kavaldjiev.** *A run-time reconfigurable Network-on-Chip for streaming DSP applications.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-02
- M. van Veelen.** *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Distributed Systems.* Faculty of Mathematics and Computing Sciences, RUG. 2007-03
- T.D. Vu.** *Semantics and Applications of Process and Program Algebra.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04
- L. Brandán Briones.** *Theories for Model-based Testing: Real-time and Coverage.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05
- I. Loeb.** *Natural Deduction: Sharing by Presentation.* Faculty of Science, Mathematics and Computer Science, RU. 2007-06
- M.W.A. Streppel.** *Multifunctional Geometric Data Structures.* Faculty of Mathematics and Computer Science, TU/e. 2007-07
- N. Trčka.** *Silent Steps in Transition Systems and Markov Chains.* Faculty of Mathematics and Computer Science, TU/e. 2007-08
- R. Brinkman.** *Searching in encrypted data.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09
- A. van Weelden.** *Putting types to good use.* Faculty of Science, Mathematics and Computer Science, RU. 2007-10
- J.A.R. Noppen.** *Imperfect Information in Software Development Processes.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11
- R. Boumen.** *Integration and Test plans for Complex Manufacturing Systems.* Faculty of Mechanical Engineering, TU/e. 2007-12
- A.J. Wijs.** *What to do Next?: Analysing and Optimising System Behaviour in Time.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13
- C.F.J. Lange.** *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML.* Faculty of Mathematics and Computer Science, TU/e. 2007-14
- T. van der Storm.** *Component-based Configuration, Integration and Delivery.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-15

- B.S. Graaf.** *Model-Driven Evolution of Software Architectures.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16
- A.H.J. Mathijssen.** *Logical Calculi for Reasoning with Binding.* Faculty of Mathematics and Computer Science, TU/e. 2007-17
- D. Jarnikov.** *QoS framework for Video Streaming in Home Networks.* Faculty of Mathematics and Computer Science, TU/e. 2007-18
- M. A. Abam.** *New Data Structures and Algorithms for Mobile Data.* Faculty of Mathematics and Computer Science, TU/e. 2007-19
- W. Pieters.** *La Volonté Machinale: Understanding the Electronic Voting Controversy.* Faculty of Science, Mathematics and Computer Science, RU. 2008-01
- A.L. de Groot.** *Practical Automaton Proofs in PVS.* Faculty of Science, Mathematics and Computer Science, RU. 2008-02
- M. Bruntink.** *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03
- A.M. Marin.** *An Integrated System to Manage Crosscutting Concerns in Source Code.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04
- N.C.W.M. Braspenning.** *Model-based Integration and Testing of Hightech Multi-disciplinary Systems.* Faculty of Mechanical Engineering, TU/e. 2008-05
- M. Bravenboer.** *Exercises in Free Syntax: Syntax Definition, Parsing, and Assimilation of Language Conglomerates.* Faculty of Science, UU. 2008-06
- M. Torabi Dashti.** *Keeping Fairness Alive: Design and Formal Verification of Optimistic Fair Exchange Protocols.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07
- I.S.M. de Jong.** *Integration and Test Strategies for Complex Manufacturing Machines.* Faculty of Mechanical Engineering, TU/e. 2008-08
- I. Hasuo.** *Tracing Anonymity with Coalgebras.* Faculty of Science, Mathematics and Computer Science, RU. 2008-09
- L.G.W.A. Cleophas.** *Tree Algorithms: Two Taxonomies and a Toolkit.* Faculty of Mathematics and Computer Science, TU/e. 2008-10
- I.S. Zapreev.** *Model Checking Markov Chains: Techniques and Tools.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11
- M. Farshi.** *A Theoretical and Experimental Study of Geometric Networks.* Faculty of Mathematics and Computer Science, TU/e. 2008-12
- G. Gulesir.** *Evolvable Behavior Specifications Using Context-Sensitive Wildcards.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13
- F.D. Garcia.** *Formal and Computational Cryptography: Protocols, Hashes and Commitments.* Faculty of Science, Mathematics and Computer Science, RU. 2008-14
- P. E. A. Dürr.** *Resource-based Verification for Robust Composition of Aspects.* Faculty of Electrical Engineering,

Mathematics & Computer Science, UT. 2008-15

E.M. Bortnik. *Formal Methods in Support of SMC Design.* Faculty of Mechanical Engineering, TU/e. 2008-16

R.H. Mak. *Design and Performance Analysis of Data-Independent Stream Processing Systems.* Faculty of Mathematics and Computer Science, TU/e. 2008-17

M. van der Horst. *Scalable Block Processing Algorithms.* Faculty of Mathematics and Computer Science, TU/e. 2008-18

C.M. Gray. *Algorithms for Fat Objects: Decompositions and Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-19

J.R. Calamé. *Testing Reactive Systems with Data - Enumerative Methods and Constraint Solving.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-20

E. Mumford. *Drawing Graphs for Cartographic Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-21

E.H. de Graaf. *Mining Semi-structured Data, Theoretical and Experimental Aspects of Pattern Evaluation.* Faculty of Mathematics and Natural Sciences, UL. 2008-22

R. Brijder. *Models of Natural Computation: Gene Assembly and Membrane Systems.* Faculty of Mathematics and Natural Sciences, UL. 2008-23

A. Koprowski. *Termination of Rewriting and Its Certification.* Faculty of Mathematics and Computer Science, TU/e. 2008-24

U. Khadim. *Process Algebras for Hybrid Systems: Comparison and Development.* Faculty of Mathematics and Computer Science, TU/e. 2008-25

J. Markovski. *Real and Stochastic Time in Process Algebras for Performance Evaluation.* Faculty of Mathematics and Computer Science, TU/e. 2008-26

H. Kastenberg. *Graph-Based Software Specification and Verification.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-27

I.R. Buhan. *Cryptographic Keys from Noisy Data Theory and Applications.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-28

R.S. Marin-Perianu. *Wireless Sensor Networks in Motion: Clustering Algorithms for Service Discovery and Provisioning.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-29

M.H.G. Verhoef. *Modeling and Validating Distributed Embedded Real-Time Control Systems.* Faculty of Science, Mathematics and Computer Science, RU. 2009-01

M. de Mol. *Reasoning about Functional Programs: Sparkle, a proof assistant for Clean.* Faculty of Science, Mathematics and Computer Science, RU. 2009-02

M. Lormans. *Managing Requirements Evolution.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-03

M.P.W.J. van Osch. *Automated Model-based Testing of Hybrid Systems.* Faculty of Mathematics and Computer Science, TU/e. 2009-04

- H. Sozer.** *Architecting Fault-Tolerant Software Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-05
- M.J. van Weerdenburg.** *Efficient Rewriting Techniques.* Faculty of Mathematics and Computer Science, TU/e. 2009-06
- H.H. Hansen.** *Coalgebraic Modelling: Applications in Automata Theory and Modal Logic.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-07
- A. Mesbah.** *Analysis and Testing of Ajax-based Single-page Web Applications.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-08
- A.L. Rodriguez Yakushev.** *Towards Getting Generic Programming Ready for Prime Time.* Faculty of Science, UU. 2009-9
- K.R. Olmos Joffré.** *Strategies for Context Sensitive Program Transformation.* Faculty of Science, UU. 2009-10
- J.A.G.M. van den Berg.** *Reasoning about Java programs in PVS using JML.* Faculty of Science, Mathematics and Computer Science, RU. 2009-11
- M.G. Khatib.** *MEMS-Based Storage Devices. Integration in Energy-Constrained Mobile Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-12
- S.G.M. Cornelissen.** *Evaluating Dynamic Analysis Techniques for Program Comprehension.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-13
- D. Bolzoni.** *Revisiting Anomaly-based Network Intrusion Detection Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-14
- H.L. Jonker.** *Security Matters: Privacy in Voting and Fairness in Digital Exchange.* Faculty of Mathematics and Computer Science, TU/e. 2009-15