# Guess My Vote: A Study of Opacity and Information Flow in Voting Systems

Thesis by

Thea Peacock

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

University of Newcastle upon Tyne
Newcastle upon Tyne, UK

2006

NEWCASTLE UNIVERSITY LIBRARY
205 36578 2
Thesis L8341

To Motivation and Inspiration.

# Acknowledgements

Extreme gratitude to my supervisor Professor Peter Ryan, for all his help and advice, without which a thesis would not have been possible. I also thank Dr. Jeremy Bryans and Dr. Lindsay Marshall for their support, and Dr. Einar Vollset for letting me use his LaTeX [4] template. Partial funding of this work was provided by DSTL [1] and the EPSRC [2] DIRC [3] project.

# Abstract

With an overall theme of information flow, this thesis has two main strands. In the first part of the thesis, I review existing information flow properties, highlighting a recent definition known as *opacity* [25]. Intuitively, a predicate $\phi$ is opaque if for every run in which $\phi$ is true, there exists an indistinguishable run in which it is false, where a run can be regarded as a sequence of events. Hence, the observer is never able to establish the truth of $\phi$. The predicate $\phi$ can be defined according to requirements of the system, giving opacity a great deal of flexibility and versatility.

Opacity is then studied in relation to several well-known definitions for information flow. As will be shown, several of these properties can be cast as variations of opacity, while others have a relationship by implication with the opacity property [139]. This demonstrates the flexibility of opacity, at the same time establishing its distinct character.

In the second part of the thesis, I investigate information flow in voting systems. Prêt à Voter [36] is the main exemplar, and is compared to other schemes in the case study. I first analyse information flow in Prêt à Voter and the FOO scheme [59], concentrating on the core protocols. The aim is to investigate the security requirements of each scheme, and the extent to which they can be captured using opacity.

I then discuss a systems-based analysis of Prêt à Voter [163], which adapts and extends an earlier analysis of the Chaum [35] and Neff [131], [132], [133] schemes in [92]. Although this analysis has identified several potential vulnerabilities, it cannot be regarded as systematic, and a more rigorous approach may be necessary. It is possible that a combination of the information flow and systems-based analyses might be the answer.

The analysis of coercion-resistance, which is performed on Prêt à Voter and the FOO scheme, may exemplify this more systematic approach. Receipt-freeness usually means that the voter is unable to construct a proof of her vote. Coercion-resistance is a stronger property in that it accounts for the possibility of interaction between the coercer and the voter during protocol execution. It appears that the opacity property is ideally suited to expressing the requirements for coercion-resistance in each scheme. A formal definition of receipt-freeness cast as a variation of opacity is proposed [138], together with suggestions on how it might be reinforced to capture coercion-resistance.

In total, the thesis demonstrates the remarkable flexibility of opacity, both in expressing differing security requirements and as a tool for security analysis. This work lays the groundwork for future enhancement of the opacity framework.

# Contents

# List of Figures

# Chapter 1

# Introduction

The main theme of this thesis is information flow, with a special emphasis on information flow in voting systems and its effect on their security requirements. The work contained has two major strands. In the first part, I concentrate on information flow properties, and, in particular, a fairly recent property known as opacity [25], [26], [27]. Intuitively, a predicate, $\phi$, is opaque if for every run in which $\phi$ is true, there exists an indistinguishable run in which it is false, where a run can be regarded as a sequence of events.

The notion of opacity may have sprung from the framework of Hughes et al, in which partial knowledge of a system is described by "opaqueness" under various "function views" [84]. Mazaré later used the term "opacity" intuitively, to describe the inability of an observer to distinguish between two messages [108]. Opacity has since been formalised by Bryans et al, and used to model information flow in various situations [25], [26], [27]. However, the opacity framework still offers ample opportunities for further research.

In the definition above, $\phi$ can represent any property of concern, giving opacity the ability to capture the security requirements of a wide range of different applications. It is therefore a highly useful property to apply where there are varying and stringent demands on security, such as in voting systems. As security requirements may vary depending on the particular voting protocol, there is a need for flexibility.

From an investigation on the relationship between opacity and other existing definitions of secure information flow [139], it appears that variations of opacity can, in fact, be mapped directly to certain properties such as non-deducibility [178], while there is a relationship by implication to others, such as non-interference [66]. In each case, the predicate, $\phi$, has to be defined in a suitable way, for example, with non-leakage [185], opacity is expressed over the initial states of the system. This will be clarified in Chapter 3.

In the second part of the thesis, the security properties of voting systems are analysed in three different ways. The overall aim is to apply opacity in a practical situation, and to demonstrate its extreme flexibility. Prêt à Voter [36] is used as the main exemplar, but to provide a comparison, the

Chaum [35], Neff [131], [133], [132] and FOO [59] schemes are also analysed. Prêt à Voter is derived from the Chaum scheme [35], but the ballot forms are constructed using different cryptographic techniques, and voter interaction with the system is much simpler. Further detail on all four schemes is provided in Chapter 4.

There is first an analysis of the information flow in both Prêt à Voter and the FOO scheme. The FOO scheme differs markedly from Prêt à Voter in a number of ways, and has rather different security requirements. As will be seen later, many of the information flow requirements in both systems can be expressed as variations of opacity. Another interesting, but perhaps incidental, result is that safety and liveness properties appear to be more appropriate than information flow properties for stating many of the requirements in both schemes. Informally, a safety property stipulates that a "bad thing" does not happen in any execution [97], [172], and liveness that a "good thing" eventually happens [97], [12]. Safety properties have been identified as being "enforceable", distinguishing them from liveness and information flow properties, which cannot be enforced [172]. There is a further discussion on this topic in Chapter 2.

The second form of analysis extends the work of Karlof et al [92], in which the Chaum and Neff schemes are analysed from the perspective of a system considered in its entirety. They identify a number of vulnerabilities which can arise due to interactions between the various components of each protocol, such as hardware, software, voting officials and the voters themselves. Chapter 6 describes a similar analysis of Prêt à Voter [163], which has likewise provided elucidating results.

This differs from the information flow analysis in that there is a greater emphasis on interaction with the environment. Furthermore, information flow is studied at a higher level than previously. Although highly useful, it may not be sufficiently methodical, and there may be doubts that all vulnerabilities in a system have been identified. It is possible that a combination of the two analyses may constitute a more systematic security analysis. For example, both forms of analysis identify chain-voting as a potential vulnerability in Prêt à Voter, but at different levels; the first by examining the core protocol, and the other by focussing on implementational issues. In the chain-voting attack, an outsider obtains an unused ballot form, marks his vote choice and persuades a voter to cast it at the polling station. If the voter returns with a fresh ballot form, the process can be repeated with another voter. A useful side result is that the systems-based analysis has generated a "taxonomy of vulnerabilities", which may act as a guideline for further development. This is discussed in Chapter 6.

Coercion-resistance, usually defined as the inability of a voter to prove her vote to a third party, is an important property of voting systems. Juels et al [91] have extended the definition to include resistance to randomisation, simulation and forced abstention. These terms will be explained in Chapter 7, in which both Prêt à Voter and the FOO scheme are analysed for coercion-resistance [138]. The primary aim is to capture coercion-resistance in terms of a suitable form of opacity, and hence, to build on the results of the previous two analyses.

As mentioned above, Prêt à Voter is potentially vulnerable to chain-voting, which could be regarded as a form of coercion. Otherwise, it has a high degree of resistance to coercion arising from the subliminal channels identified in [92], which can occur through random choices made during ballot form creation, or if there are alternative valid representations of the vote. However. kleptographic channels are a possible threat [65]. These attacks are described in further detail later in the thesis. The FOO scheme is open to coercion, and provides an interesting comparison in the analysis. Confirming initial intuition, it appears that variations of opacity can be used to informally express coercion-resistance in a succinct way. A formalisation of receipt-freeness, a weaker property, is proposed [138], and the way in which this definition might be extended to capture coercion-resistance is discussed. There is further detail on coercion-resistance and receipt-freeness in Chapters 6 and 7.

Designated Verifier Proofs [88] and deniable encryption [30] feature in the literature on voting systems, the former perhaps having more usefulness in protocol design [91], [82], [87], [38]. It appears that it may also be possible to cast these properties in terms of a suitable form of opacity [138]. This is also discussed in Chapter 7.

In total, this work indicates that opacity is a remarkably flexible and highly useful property, both for modelling information flow and as a tool for analysing secure systems. Not wishing to stray further into concluding remarks, I now summarise the work on which my thesis is based. There are two main parts of the thesis. In the first, I investigate the opacity property, in particular to find any relationships it may have to existing definitions of secure information flow [139]. In the second, I carry out a case study of voting systems, testing the usefulness of the opacity property in a practical situation. This was done by way of a series of analyses on some recent voting schemes. Overall, the aim is to test the flexibility of the opacity property, and to show that its flexibility is an advantage in the specification and verification of secure systems.

The thesis is organised as follows. In Chapter 2, I present a literature review which provides a background for the ensuing work. Important and interesting work on information flow to date is discussed. Included in this chapter is a review of literature on anonymity, which could be regarded as a specialised form of information flow, in that it concerns the hiding of identity. Anonymity is typically a requirement in voting schemes, and hence, its relevance to the thesis. a definition of anonymity cast as opacity is the subject of future research. Defining different forms of anonymity in terms of opacity is the subject of future work. In Chapter 3, formal definitions opacity are given, followed by an investigation of its relationship to several existing security properties, such as non-interference and non-deducibility [139].

The case study begins in Chapter 4, with a general overview of voting systems and a discussion of some recent work in the field. This is followed by overviews of Prêt à Voter, and the Chaum, Neff and FOO schemes, which feature in the case study. An information flow analysis of Prêt à Voter and the FOO scheme is presented in Chapter 5, followed by a systems-based analysis of Prêt à Voter in Chapter 6 [163]. Coercion-resistance and receipt-freeness are discussed in Chapter 7, together with an analysis of coercion-resistance in Prêt à Voter and the FOO scheme. This leads to a formal definition of receipt-freeness cast as a variation of opacity [138]. Finally, in Chapter 8, I discuss possible future work and present my conclusions.

# Chapter 2

# Secure Information Flow - A Review of Literature

## 2.1 Introduction

The security of a system can depend on many factors, for example, the operational environment and interactions with other systems. Whether or not the system is "secure" can also depend on the definition of "security", the properties required of the system, and how well the requirements are satisfied. System security typically involves preserving confidentiality, integrity and availability. In the case of confidentiality and integrity, this requires the prevention of illegal flow of information.

Confidentiality has been defined in various ways, for example, secrecy, anonymity or privacy. Usually, secrecy refers to data, anonymity to identity, and privacy to relationships between entities [84], but they are all concerned with hiding information. By convention, protected information is usually assigned a high security level, and confidentiality requires that requires that information should not flow from high to low. With integrity information should not flow from an unauthorised level to high, and could, roughly speaking, be regarded as the dual of confidentiality [23].

Information flow is often considered to be of two types: explicit and implicit. The flow is explicit when information is passed directly between processes. Implicit flows are more subtle in that information is transferred indirectly, that is, via covert channels or inference. Analysing even simple systems for either type of flow is difficult, and this is particularly true of implicit flows. This problem is discussed in more detail later in the chapter.

Information flow security is notoriously difficult to define. In fact, it has been a subject of security research for over thirty years, and, although many definitions have been proposed, there is as yet, no universally accepted solution. This is no doubt due to the great diversity of systems and the sheer complexity of interactions with other systems and the environment. Security requirements may also vary widely depending of the system under consideration. With the rapid increase in the complexity of computer systems and networked communications, a definition of security is likely to become even more elusive.

This chapter provides an overview of the vast amount of research in the field to date. Research on information flow properties is, where possible, put into historical context, and serves as a basis for investigating possible links between opacity and other information flow properties. The focus is on confidentiality properties in possibilistic abstract systems, since they are perhaps more relevant to the thrust of this thesis. However, in order to provide an interesting comparison, mention will also be made of language-based information flow security and probabilistic models. In any case, it is desirable to have a thorough understanding of both approaches, and to be aware of the arguments for and against each one of them. I return to this issue later in the chapter.

In their seminal paper, Goguen et al [66] point out that it is necessary to verify that a specification is satisfied by both a security policy and system code. This would appear to imply that theoretic and language-based models both have a role in providing security assurance, and that perhaps each one impacts on the other. Although interesting, this consideration is beyond the present scope. The term "models" can be used in the sense described by McLean [115], that is, an interface between a mechanism and a system specification. A security model may place restrictions on system inputs and outputs, but does not specify how the restrictions should be enforced. It is merely a guideline for ensuring that the system meets certain security requirements. A "security policy" in the terminology of Goguen et al [66] could thus be regarded as a model. At the theoretical level, a security model specifies the conditions under which information may or may not flow through the system.

Sabelfeld et al [166] give various examples of why security mechanisms fall short of security requirements. For example, access control may enforce restrictions on how confidential information is accessed, but not how it is propagated. In the language-based context, a type system is used to ensure that program code satisfies a given security condition or policy. Static analysis is often performed to check all possible execution paths, and so ensure that the code meets the system specification. This could be regarded as merging the enforcement mechanism with the security model, and perhaps explain why most type systems and security conditions are compositional [166]. The same is not generally true of abstract security models. Much work has been done in the field of information flow, with the aim of achieving compositionality, as will be discussed later.

This chapter is structured as follows. In Section 2.2, access control is discussed. I concentrate on the well-known Bell and LaPadula model [20], [19], since it is often regarded as the precursor of later security models. Following this in Section 2.3 is a discussion on covert channels, which were identified as a problem in early access control models, and prompted subsequent research in theoretical definitions. Several of the well-known, and some perhaps less-well-known transitive and intransitive information flow properties are then discussed in Sections 2.4 and 2.5 respectively, together with an explanation of the terms "transitive" and "intransitive". I also attempt, where possible, to indicate any relationships which may exist between them. Opacity is mentioned only briefly here, as a detailed description of the property is provided in Chapter 3. In Section 2.6, I describe several general frameworks for security properties that have been proposed.

Section 2.7 examines modelling of information flow: the different formalisms that have been used to express security properties, compositionality and the key aspects of the debate on possibilistic versus probabilistic models, together with some relevant work in each area

Anonymity can also be regarded as an information flow property, but of a more specific type in that it usually concerns identity. In Section 2.8, existing work on anonymity is discussed: the various ways in which it has been formalised and implemented, along with pseudonymity, which is a weaker form of anonymity. Finally there is a summary of the work presented.

## 2.2 Access Control

Many of the early security models were based on the principles of access control, which emulated access to information in the pen and paper world. As a typical example, a file may or not be released, depending on the requesting agent's security clearance. It is not difficult to see how this model has been abstracted into the computer environment. Information is usually classified at various levels within a hierarchy, and for permission to be granted, a given security clearance would have to satisfy the security policy of the organisation concerned.

This is in fact the basis of Multi-level Security (MLS), which is often implemented in the military. File classifications might be, for example, Top Secret, Secret, Confidential or Unclassified, arranged linearly in a lattice. As regards access to information, an agent's clearance would have to be at least equal to that of the file for read permission to be granted. The converse applies to write permission.

The lattice could also be non-linear, for example

Top Secret
Secret-Terrestrial | Secret-Space
Confidential
Unclassified

Figure 2.1: Security Classification - Non-linear Lattice

According to a scheme such as the one in Figure 2.1 above, information could be described as compartmentalised, and file permissions were usually exercised according to the *principle of least privilege* [169] or *need-to-know*. For example, Secret-Terrestrial and Secret-Space classifications would be mutually exclusive, and so an agent with Secret-Terrestrial clearance would not be able to read Secret-Space information, and vice versa.

With paper-based documents, these rules could be subverted fairly easily. Information could be exchanged illegally between individuals, either directly, perhaps away from the workplace, or indirectly, such as via pre-arranged signals. Similarly, but not quite analogously, when translated into computer systems, there can be information leaks between interacting processes. This can be mitigated by ensuring *end-to-end security* [166] at the program level, or, more abstractedly, by theoretical models and mathematical definitions. Some of these methods will be discussed in the ensuing sections.

One of the earliest access control models is based on the *Access Matrix (ACM)*, designed by Lampson [98]. The set of subjects, $S$, objects, $O$, $S \subseteq O$, and privileges, $P$, are stored in a matrix, $M$, such that each cell of $M$, $[s, o]$, $s \in S$, $o \in O$ corresponds to the access rights, $(p_1, \ldots, p_n)$, $p_i \in P$, $0 < i \leq n$, of subject $s$ to object $o$. However, with large, sparse matrices, the ACM can be wasteful of memory space and more practical implementations were devised. Non-empty rows of the ACM can be stored as an authorisation table, by row as Access Control Lists (ACLs), or by column as capabilities. These are found in current systems for access control. A good general reference on the subject is [170].

There are two basic forms of access control. With *Mandatory Access Control (MAC)*, subjects are issued privileges on objects by a central authority. With *Discretionary Access Control (DAC)*, subjects may be allowed to change the privileges associated with files that they own. Although a full treatment of access control is beyond the present scope, the work of Bell and LaPadula will be discussed in some detail, as it had an influence on the development of subsequent theoretical security models.

### 2.2.1 The Bell and LaPadula Model

Based on a ACM, the Bell and LaPadula model (BLP), [19], [20] is an example of both MLS and MAC. Subjects, which could be users or computer processes, and objects, for example files or other units of data, are grouped, respectively, according to security clearance and level. Information flow is controlled by the *simple security* and $*$ *(star)* properties. By the former, a subject may only have read access if the security clearance dominates that of the object. This is also known as *no read up*. The star property states that a subject may only have write access if the security clearance is dominated by that of the object, which is also known as *no write down.*

According to the *Basic Security Theorem (BST)* of the BLP model, a system is secure if it is both state- and transition-secure. However, a major weakness is that apart from preserving state security, there are no other restrictions on transitions. The problem is demonstrated by McLean's "System Z" [116], which has a secure initial state and only one transition. Any access request by any subject results in all objects being downgraded to the lowest level. Despite satisfying the BST, the system is clearly insecure.

Other flaws with the BLP model were also found. McLean [114] showed that due to lack of formal definition, the usual meanings of "read" and "write" could be inverted and again lead to an insecure system. The BLP model can also be overly restrictive. For example, the so-called *tranquility* *principle* states that an object's classification should remain unchanged. In practice, objects may need to be re-classified, such as in the downgrading of information after a certain period of time.

In a later revision of the model, subjects could be assigned a "current level". This could be any level, provided that it is dominated by the subject's clearance. However, a "trusted subject" would be required to perform these level assignments, and this can be difficult to implement in a secure way.

## 2.3 Covert Channels

Perhaps a more serious problem with the BLP is its vulnerability to covert channels, which provide a means for illegal exchange of information. Covert channels were first defined by Lampson as "those not intended for information transfer at all" [99]. Consider, for example, an error response to a low level request for a high level file. The low level requesting agent may be able infer certain information about the existence or non-existence of the file. Cuppens [40] gives an example of this type of implicit flow in a multi-level database.

Lampson [99] shows that a variety of covert channels are possible depending on how system signals are exploited. Foley discusses timing channels [57]. Probabilistic channels are studied by Moskowitz in [124]. Sabelfeld et al [166] give a comprehensive overview of several different channel types in the language-based context. Karlof et al [92] describe possible subliminal channel attacks in voting schemes where ballot forms are created on demand. By careful choice of random values during the interaction between the voter and voting device, messages can be encoded in ballot receipts. Related to this are kleptographic channel attacks, where information can be leaked through a trapdoor built into system code, and in a way that is difficult to detect [65].

Covert channels can arise from the mapping of "subjects" and "objects" to components of a computer system, possibly exacerbated by resource sharing [115], [160]. They are usually identified by covert channel analysis [113] and either removed, or limited to some acceptable level, for example, by introducing "noise" to lower the channel capacity [111], [125].

McCullough has studied various covert channels and shown that, for example, positive half-bit channels can be converted to negative, and that covert channels can be introduced or magnified by system composition [111]. In McCullough's model, input from a Trojan horse can result in "yes", "no" or "maybe" signals. A positive half-bit channel, for example, is one where the observer can be sure that there has been a "yes" response, and there is no indication that there has (possibly) not been an input, that is, a "maybe" signal.

In language-based security, implicit flows are located by static program analysis or type-checking [166]. However, detecting and correcting covert channels at the end of the development process can be expensive, and it would be preferable to prevent their occurrence at the design stage [115].

In the ensuing sections, I discuss a different approach to information security. Based on preventing non-secure flows by mathematical constraints on system events, it is in direct contrast with the lack of formal semantics in the BLP model.

## 2.4 Transitive Information Flow Properties

There are many existing definitions for secure information flow and it would be difficult to describe them all. This, and the ensuing sections survey important and interesting work, where possible placing them in chronological order, but also mentioning related or incidental work at the same time. The aim is to chart the progress made in this area so far.

In this section I discuss transitive flow properties, which usually analyse potential flow between two sets of users: High and Low [152]. Transitive flow can be represented by the following relation [152]

$$A \leadsto B \wedge B \leadsto C \Rightarrow A \leadsto C$$

where $\leadsto$ is the information flow relation.

Intuitively, following the general rule that $High \not\leadsto Low$, $A$ has a lower level than $B$ and $B$ a lower level than $C$. However "this argument misses the crucial possibility that some high-level users are trusted to downgrade material or otherwise influence low-level users" [152], which is the essence of intransitive flow as discussed in the next section.

### 2.4.1 Non-interference

Drawing on earlier work by Cohen [39] and Feiertag [52], Goguen et al's *non-interference* is one of the first attempts at addressing the problem of covert channels [66], [67]. Based on a deterministic state machine, non-interference is defined in [67] as follows.

$$G, A :| \ G', B$$

where $G, G'$ are groups of users, and $A, B$ are sets of state transition commands. ":|" can be thought of as a "one-way mirror", such that G can possibly see G', but G' cannot see G. In other words, ":" and "|" are respectively the "open" and "closed" sides of the mirror.

Thus, $G$ does not interfere with $G'$ if

$$out([[w]], v, r) = out([[P_{G,A}(w)]], v, r)$$

where $out(s, u, r)$ is a function that gives the result of user, $u$, executing a command, $r$, in the state, $s$, and the *purge* function, $P_{G,A}(w)$, gives the sequence given by eliminating all the pairs with $u \in G$ and $c \in A$. In other words, if $G$ denotes High-level users and $G'$, Low, then in the space of possible system traces, High inputs cannot interfere with any outputs to Low.

For complete elimination of covert channels, probability and time are sometimes thought to be important considerations in modelling information flow. Probabilistic non-interference is discussed later in the chapter. McLean suggests that time should be considered as part of system inputs and outputs [115].

Goguen et al also introduced the notion of *unwinding* [67]. The idea is to replace quantification over all possible traces with a conditional relation on single state transitions, which makes the verification process more tractable. In [164], Ryan discusses unwinding for definitions of non-interference in *CSP* [83], and proposes an unwinding relation for the non-deterministic case in terms of *power bisimulation* [62]. Two processes are said to be strongly bisimilar if the events of one matches every event, including internal events, $\tau$, of the other. Weak bisimulation allows for arbitrary interleaving of the *taus*. With power bisimulation, a relation is drawn between two bisimilar processes on sets of states, rather than individual transitions. As in weak bisimulation, this abstracts away details on internal transitions. For full details, the reader is referred to [160]. Mantel gives unwinding rules for possibilistic security properties which he relates to his *Basic Security Predicates (BSPs)* [103], [102]. His work is discussed later in the chapter.

Non-interference was later extended to enable conditional flows [67], which increased its flexibility. In its original form, it can sometimes be too strong. An example is a cryptographic system [115] where Low may observe an encrypted High information stream, but, if he does not learn any protected information, the system could still be regarded as secure. Ryan et al have proposed a form of generalised non-interference which can be adapted to model, for example, encryption and probability [164]. Conditional and intransitive information flow are discussed further shortly.

Another objection to the original form of non-interference is that it is specific to deterministic systems. Several modifications of non-interference have been proposed, with the aim of defining a more general security property. Some of these definitions will be discussed in this chapter. There have also been recent process algebraic formulations of non-deterministic non-interference, for example, in [55], [158], [164].

Sabelfeld et al express non-interference for programs as a condition for security type-checking: that a variation of High input should not cause a variation of Low output [166].

## 2.4.2 Non-deducibility

Sutherland introduced *non-deducibility* [178] to capture non-determinism by the requirement that any Low observation should be compatible with any acceptable High input [115]. It was originally defined as follows.

"Given a set of possible worlds $W$ and two functions $f1$, $f2$ with domain $W$, information flows from $f1$ to $f2$ if and only if there exists some possible world $w$ and some element $z$ in the range of $f2$ such that $z$ is achieved by $f2$ in some possible world but in every possible world $w'$ such that $f1(w') = f2(w)$, $f2(w')$ is not equal to $z$" [178].

Another way of stating this, in the style of, for example [164], [42], is that for two users, High, $H$, and Low, $L$, and for all possible traces of the system, $t, t' \in \tau S$, there exists a trace $t''$ such that

$$(t \upharpoonright H = t'' \upharpoonright H) \wedge t'(\upharpoonright L = t'' \upharpoonright L)$$

.

The trace, $t''$ is thus compatible with both High inputs in $t$ and Low's observation of $t'$.

McCullough noticed a weakness in non-deducibility, due to the lack of constraints on the interleaving of High inputs and Low events in a trace compatible with both [110]. He also showed that neither non-deducibility nor non-interference were compositional, and proposed *Hook-up Security*, discussed later, which is stronger than non-deducibility and preserved under certain types of system composition.

Focardi et al have defined non-deducibility in *Security Process Algebra*, named *Bisimulation-based Non-Deducibility on Compositions, (BNDC)* [55], which is also non- compositional. However, they show that their property *Strong Bisimulation Strong Non-deterministic Non-Interference(SBNNI)* is a sufficient condition for *BNDC* and that it does hold under composition. In [56], BNDC is used to analyse cryptographic protocols. Ryan has also shown that his *CSP* definition of non-deterministic non-interference is compositional [160]. In [175] Schneider discusses non-interference formulations based on may- and must-testing equivalences, and shows that while the former is compositional, this is not true of the latter.

Another problem with non-deducibility, first identified by Guttman et al [75], is that, like hook-up security, it is only concerned with High inputs. This was attributed to the reasoning that High output is derived from High input, Low input and system events, and that Low cannot learn anything significant from High output without knowledge of High input. However, Guttman et al show this view to be false. Their definition of *non-deducibility on outputs* states that for any two traces, $t, t'$ of a system, $S$, there exists a trace, $t''$ such that $t''$ contains the same Low events as $t$ and the same High events as $t'$. In a weakened version, internal events are also hidden [75].

Wittbold et al [186] were also of the opinion that non-deducibility on High inputs was too weak, and proposed *non-deducibility on strategies*. They define a (High or Low) strategy as a series of functions that determine the user's next input from previous inputs and outputs. A system is non-deducible on strategies if, for any Low observation and any strategy, there is a trace compatible with both. Their work is also concerned with the inadequacy of possibilistic models against statistical inference [186]. They demonstrate that conventional methods, such as increasing noise to limit covert channel capacity, are ineffective. Although they argue that probabilities should be taken into account, they do not extend their own work in this way.

Non-deducibility has also been criticised for being a symmetric property [117]. However, Halpern et al [78] argue that this is not the case, and that it depends on the choice of the information functions. For example, the function $f$ might prevent flow from $g$ to $f$, but this says nothing about flow in the reverse direction. In fact, Sutherland makes a similar comment himself in [178].

### 2.4.3 Non-inference

Another property applicable to non-deterministic systems is *non-inference*, proposed by O'Halloran in [134]. The original CSP formulation is as follows.

"Whenever $A \cap B = \{\}$
$A \not\rightsquigarrow^{P} B \iff \forall t : \tau P \cdot (t \setminus A \in \tau P) \vee (t \upharpoonright B)$" [134].

Intuitively, "any behaviour of a system $P$, either has a corresponding behaviour involving no interactions through window $A$ or involves no interactions through window $B$. If $P$ has a corresponding behaviour with all interactions through $A$ removed, then these behaviours viewed through $B$ will be indistinguishable, assuming $A$ and $B$ to be disjoint windows" [134]. This means that $B$ (Low) cannot eliminate the possibility that $A$ (High) did nothing. As noted by McLean [118], this captures non-determinism with the possibility of High output without High input. However, McLean also observes that for deterministic systems, non-inference is equivalent to non-interference [118]. This appears somewhat unclear, as non-interference is a stronger property than non-inference in that it "requires all behaviours of $P$ to have a corresponding behaviour of $P$ with all interactions through $A$ removed" [134]. The disjunct in the definition above "allows the possibility of information flow from $A$ to other disjoint windows distinct from $B$" [134].

These observations should hopefully become clearer when I relate non-inference to opacity in Chapter 3 [139].

## 2.5 Intransitive Information Flow Properties

As mentioned earlier, the BLP model and non-interference are often criticised for being too rigid, in that information flow between High and Low is strictly prohibited. In practical systems, there are situations where the rule may need to be relaxed, for example, when encrypted secret information passes through a public channel, information is purchased, or average figures from a confidential database are released for statistics. This is sometimes referred to as *declassification* or down-grading. that is, the security level of confidential information is lowered so that it can be publicly accessed.

Declassification could be regarded as partial or selective information release, as, for example, in [39], [67]. In [76], Haigh et al develop a version of conditional non-interference for a multi-domain system.

In [104], Lowe presents a model whereby permitted information flow is quantified. An "acceptable" covert channel capacity is defined in terms of the number of bits that High can transmit to Low and the rate at which information can be transmitted. These ideas are related to Ashby's information transmission [14], that is, when there is variety conveyed from source to destination, and Shannon's information theory [177], that concerns the amount of information transmitted. Other work in this area which draws on information theory is found in [57], [186]. For modelling the rate of flow, Lowe found that discrete time CSP [174], gives more tractable results than Timed CSP [174], [154].

The declassification process can also be thought of as a "trusted channel" between High and Low. Intransitive flow has been modelled in this way, using various methods of physical separation between High and Low, for example in [156], [44], [152]. Other methods also specify where information can be released, but at an abstract level. An example is in *constrained non-interference* [164], where information can flow via a *constrain* process.

In [127] and [24] downgrading is represented by unwinding rules that make use of modified purge functions. Both provide proofs for compositionality, the latter also includes proofs for refinement. Mantel represents intransitive non-interference [106] as exceptions to a standard flow policy. The policy is based on a MLS lattice, and a security condition controls where downgrading can occur.

### 2.5.1 Dynamic Level Assignment

In another approach, information flow is dependent on factors such as time, location or identity of the requesting agent, and is sometimes known as *dynamic level assignment*. In [179], Sutherland et al present a version of non-deducibility with dynamic level assignments. Their model consists of a synchronous state machine, the security of which is defined in terms of two functions. One returns the levels (either High or Low) associated with *event occurrences*, and the other, *downgrade*, returns either High or Low depending on the time by the clock at the final state of a trace extending an event occurrence.

Synchronous dynamic level assignment of both subjects and objects has also been used to model causality in [44]. This is achieved by two sets of conditions. The first specifies the current level, $c$, of subject, $s$. The second, an object's classification, which is determined by a function, $l$. A system is secure, that is, satisfies causality, with respect to $s$ if $c$ and $l$ are securely defined. Cuppens et al show that this technique can be applied to systems with shared resources. However, the problem that evolving object classifications ultimately reach a plateau is not addressed.

### 2.5.2 Rushby's Intransitive Non-interference

In his seminal paper [156], Rushby defines an intransitive flow policy, denoted by $\rightsquigarrow$, and a set of unwinding conditions that must hold for a system, $S$, to be secure with respect to $\rightsquigarrow$. The unwinding conditions, based on a modified purge function, *ipurge*, ensure that a user only observes inputs and outputs from permitted domains and for all transitions of $S$.

Rushby compares *intransitive non-interference* to non-interference, as defined by Goguen et al [66], and the BLP model, and shows that the intransitive case can be reduced to the transitive. However, like that of Goguen et al, Rushby's work is limited to deterministic systems.

A slightly unusual approach is that of Pinsky [145], in which intransitive non-interference is generalised using the theory of *absorbing covers*, which replace unwinding conditions with (transitive) non-interference. Pinsky also proves that computations for intransitive and transitive flows have equal complexity.

In [152], low-deterministic non-interference [153], which has been mentioned earlier, is used to derive a definition for intransitive flow. Roscoe et al show that this definition is stronger than the modified purge function *ipurge* of, for example, [67], [156]. Although, they criticise *ipurge* for its weakness, they only consider insecure flows due to system errors, rather than those caused directly by the function itself.

There is more recent work which draws on Rushby's work and also takes non-determinism into account [103], [102], [105], [185]. I discuss some of them in the following sections.

### 2.5.3 The Basic Security Predicates (BSPs)

In [103], Mantel defines *Backwards Strict Deletion (BSD* and *Backwards Strict Insertion of Admissible events (BSIA)*, which he calls the *Basic Security Predicates (BSPs)*. BSD demands that a High event does not add a possible Low observation. Conversely, BSIA prevents a High event from removing a possible Low observation. His work is based on an event system, similar to that used by McCullough [110]. BSD and BSIA also appear to resemble to McCullough's hook-up security [110] and its modifications [112] [89].

In Mantel's system, the set of High events is defined by $H = H_c \cup H_a \cup H_o$, where $H_c$ are events that must not be deduced by Low, $H_a$ are adaptable events and $H_o$, outputs. BSD and BSIA can be varied depending on which High events are specified.

The framework is then extended for intransitive flow by a set of unwinding rules which are similar to those devised by Rushby [156]. Mantel also shows that different predicates can be defined using a combination of the BSPs and the appropriate unwinding rules [102].

### 2.5.4 Non-influence

Also influenced by Rushby's work, are the properties *non-influence* and *non-leakage* [185]. In von Oheimb's model, users are grouped into domains. Non-leakage restricts a set of initially secret information to a particular domain. More precisely, the definition states that an observer, $u$, should not be able to distinguish any two states $s, s'$, if, during the course of a trace, $t$, from $s, s'$, no domains have been allowed to interfere directly or indirectly with $u$.

Non-influence is defined as being equivalent to (intransitive) non-interference and non-leakage combined [185]. Although both properties in their original form are designed for intransitive flow policies, von Oheimb also discusses the transitive cases.

Unwinding rules are defined for deterministic, non-deterministic, weak and strong versions of each of the two properties, non-influence and non-leakage. Similar to Mantel's work, the unwinding relations do not require symmetry, reflexivity or transitivity [102], [105], but must imply observational equivalence. I return to non-leakage in Chapter 3, in a discussion on its relationship to opacity [139].

### 2.5.5 Language-based Intransitive Flow Models

In addition to the exemplary review of language-based information flow in [166], Sabelfeld et al have compiled a survey of language-based declassification [167].

Although this area is outside the present scope, it is interesting to note one approach in particular, apparently absent in more abstract models. In the *decentralised label model* [128] information is secure only if released by the owner. In other words, the model is centred around *who* may release information. Another interesting concept is *robust declassification* [188], which is designed to prevent information release from abuse. Essentially, this stipulates that if a passive attacker cannot distinguish between two memories where secret information is altered, then no active attacker may do so.

Also of interest is work by Chong et al [37] on language-based information *erasure*. Erasure could be thought of as the opposite of declassification, that is, it allows less rather than more information flow. This appears to be particularly well-suited to applications in which information should not be retained beyond a certain period, such as ballots in electronic voting systems, patient information in medical systems and time-limited cryptographic keys. An erasure policy, expressed using a security type-system, ensures not only that certain information be erased, but also any information derived from it.

## 2.6 General Frameworks for Security Properties

There have been several attempts at devising a system in which other existing security properties, or even new ones, can be defined. One useful result is that a uniform method of expression sometimes facilitates comparison of security properties, for example in terms of relative strengths, from which a "taxonomy of properties" might be drawn. An example is the previously-mentioned work of Focardi et al [55]. In addition, the subtle differences between different properties sometimes become more apparent.

### 2.6.1 Selective Interleaving Functions

In [118] McLean formulates a class of functions which could be used to define several security properties. Each property is defined in terms of a synchronous trace set under one such function. An interesting consequence is that an originally asynchronous property can be considered in a synchronous setting.

A function takes either one or two traces and returns a trace purged of High inputs and/or outputs, and has specific interleavings of High and Low events. Traces are of the form

$t = \langle\langle({in_H}^1, {in_L}^1)({out_H}^1, {out_L}^1)\rangle, \langle({in_H}^2, {in_L}^2)\rangle({out_H}^2, {out_L}^2)\rangle, \ldots\rangle$, where $in_H$, $in_L$ are the input events of High and Low respectively, and $out_H$, $out_L$ the output events.

*Separability* is defined as the function $f$ such that $f(t, t') = t''$, $in_H(t'') = in_H(t)$, $in_L(t'') = in_L(t')$, $out_H(t'') = out_H(t)$ and $out_L(t'') = out_L(t')$.

It is comparable to the non-deducibility-type properties in that $f$ takes two traces, $t, t'$, and returns a trace, $t''$, that is compatible with the High events of $t$ and Low events of $t'$. However, it is stronger than non-deducibility, in that both High inputs and outputs are considered, and also specifies how High events are interleaved with Low events. McLean comments that it resembles Rushby's *separation kernel* [155], that is, independence between High and Low.

Non-deterministic noninterference, *generalised non-interference*, non-inference and *generalised non-inference* are also defined in terms of similar functions. Note that McLean refers to McCullogh's hook-up security as generalised non-interference [115]. The properties are ordered by relative strengths with separability ranked as the strongest. Compositionality and is also examined in [118], not only how each property composes with itself, but also with other properties.

### 2.6.2 Causality

Glasgow et al [64] use modal logics to define confidentiality with the formula $K_B\varphi \rightarrow R_B\varphi$, that is, if $B$ knows $\varphi$, then $B$ has the right to know $\varphi$.

Cuppens et al build this formula into a constraint on traces which they call *causality*. Their model is a synchronous event system, in which a trace is defined as a pair $(o, t)$, where $o$ is an *object* and $t$ a *time point.* The set of objects is divided between ***input, output*** and ***internal*** objects. $\varphi$ is $S$-valid if for all traces, $\varphi$ is true, and a system $S$ is secure with respect to $B$ if $K_B\varphi \rightarrow R_B\varphi$ is $S$-valid [42], [43].

They examine the relationship between causality, non-interference, non-deducibility and generalised non-interference. A causal system is necessarily deterministic, but the mappings are shown to be possible under the assumptions of input totality and only two sets of users.

In [43], they define ***non-disclosure on inputs***, which protects High inputs, and also ***non-disclosure***, which also protects High strategies. There is also a case study of inference control in multi-level databases in [41], [40], [45].

### 2.6.3 The Perfect Security Property

Zakinthinos et al [187] express a security property as a predicate that is true of every *low-level equivalent set (LLEQ)* in a system. For any trace $t$, a LLEQ is a trace with the same Low events in the same order as $t$. This theory is applied to several existing security properties. They then argue that McLean's separability is too strong, and introduce the *Perfect Security Property (PSP)*, claimed to be a weaker version which allows some information flow from Low to High.

They show that PSP is compositional, and argue that their assumption of input totality is necessary for cascade composition. Input totality, also assumed, for example, in [112] and [89], requires that inputs are always accepted by the system. Finally, they create a lattice of security properties ordered by relative strength.

Note the similarity between the PSP and opacity. Referring to the informal definition given earlier, a LLEQ could be regarded as Low's observation of a trace. If, for example a system, $S$, satisfies opacity, with respect to $\phi$, then every LLEQ must also satisfy opacity with respect to $\phi$. Another way of stating this is that $\phi$ is opaque to Low.

### 2.6.4 $f$-Secrecy

Halpern et al [78] define a predicate $\varphi$ as "secret" if agent $a$ never knows that $\varphi$ is true. In their *runs and systems framework*, a *global state*, $S = (s_e, s_1, \ldots, s_n)$, where $s_e$ maps to the environment and $s_i$ is the state of agent $i$, for $0 < i \leq n$. A *run* is a function from time to $S$, that is, it describes the events in one execution of the system. A *point* is a tuple $(r, m)$, in other words, a run $r$ at time $m$. An information function, $f$, maps an agent to its possible local states. An agent $j$ maintains *total $f$-secrecy* with respect to $i$ if $i$'s knowledge of $j$'s local state depends on $f$.

Total $f$-secrecy can be weakened by an *$i$-allowability function, $C$*, that maps the set of points in a system to subsets of points that $i$ may rule out. It may be reasonable under some circumstances, for instance, to allow $i$ to know certain facts about the system. The weaker form of total $f$-secrecy is referred to as $f$-secrecy [78].

Secrecy also has a syntactic definition which ensures that at any point $(r, m)$ Low cannot rule out any High-level state, that is, specifying what an agent is allowed to know. According to Halpern et al. the syntactic definition is useful when drawing on the reasoning power of (epistemic) logic, whereas the semantic alternative, given previously, facilitates model-checking and system specification. Their motivation is not, however, to provide an interface between two different formalisms, as in the function view framework of [84] to be discussed shortly. In fact, they argue against the necessity of such an interface.

Halpern et al compare their work to that of McLean [118] and Wittbold et al [186], in which synchronised traces are fixed infinite sequences of events. A point could be considered a finite trace, and they show that separability and generalised non-interference [118], when defined in terms of finite traces, become weaker than the properties as originally defined [77]. They also define an asynchronous version of $f$-secrecy [77], relating to the work of Zakinthinos et al [187]. This reveals a potential weakness in $f$-secrecy, which is due to a temporal dependency between High and Low events. They show that this can be corrected by *closure under interleavings*.

In [79], $f$-secrecy is used to model anonymity properties. Some of the research done on anonymity will be reviewed later in this chapter.

### 2.6.5 Opaqueness under Function Views

Hughes et al [84] define various forms of opaqueness on function views which represent partial knowledge of a system. A function view, $V$ is a triple $\langle F, I, K \rangle$, where $F$ is a function of the form $X \rightarrow Y$, and $I$, $K$ respectively denote image and kernel knowledge of $F$. A function is image opaque if $x \in X$ cannot be mapped to any value $y \in Y$, and kernel opaque if $\forall x, x' \in X$, no equalities $f(x) = f(x')$ can be inferred. Different forms of value opaqueness are also given, for example, $F$ is $n$-value opaque, if $\forall x \in X$, $| F(x) | \geq n$.

A motivation behind the function view framework was to provide an interface between a system specification and property specification, in particular, utilising the relatively greater strengths of process algebra and logics for the respective tasks. Although it has roots in epistemic logic [81], the function view belongs to domain theory [10], and Hughes et al claim that the framework can be applied to any process algebra and any logic

They present an extensive case study on anonymity, which includes pseudonymity and privacy. This is discussed further in the literature review on anonymity. Anonymity will also be discussed in later chapters of the thesis.

## 2.7 Modelling

### 2.7.1 Formalisms

The large amount of research on information flow to date is split between various different formalisms. Models have also been defined with a combination of formalisms, for example, state machines and modal logic [178], function views and trace systems [118]. This section provides only a general overview. Some of the work mentioned has been discussed in other sections of the chapter.

#### 2.7.1.1 State Transition Machines

*State transition machines* were common in early research, for example, [66], [67], [178], [110]. *Event systems* are a later, more a abstract form, as in [75], [89], [186]. These persist in more recent research [45], and are sometimes called *labelled transition systems* [25] or automata [125]. In some work, the states of the system are ignored and only the events considered, for example, the *trace systems* in [118].

#### 2.7.1.2 Petri Nets

At an even higher level of abstraction are formalisms such as *Petri Nets* [149] and process algebras, for example, *Communicating Sequential Processes (CSP)* [83] and *Calculus of Communicating Systems (CCS)* [121]. Process algebras are discussed in the next section. In [184], [29], [27], [26], security properties are analysed with Petri nets.

#### 2.7.1.3 Process Algebras

Process algebras, such as CSP and CCS, were designed to model interacting processes. The available semantics and operators enable convenient and efficient representation of non-determinism, abstraction and process composition and equivalence. Hence, reasoning about security in terms of exchanges between users of a system is greatly facilitated when represented in a process algebraic form. An additional advantage is that there is tool support for model-checking, which readily enables verification. Some of the work in this area is discussed in other sections of the chapter.

There is a sizeable amount of work based on CSP. Ryan [157] and Ryan et al [160] have provided CSP formulations of several security properties such as non-interference. Roscoe [150], [154] and Roscoe et al [151] have examined non-determinism and refinement in depth. [71], [101], [58] are also concerned with these topics. Schneider has modelled various security properties [173] in CSP. In [176] Schneider et al propose a CSP formalisation of anonymity . Focardi and Gorrieri [55] have constructed a taxonomy of security properties in *Security Process Algebra*, which is based on CCS. They also compare the relative strengths of these properties and discuss compositionality issues.

Hennessey [80] uses *Security $\pi$-Calculus*, a typed version of *$\pi$-Calculus* [122], in his study of non-interference.

In [151], Roscoe et al present an interesting comparison between CCS, which is strongly operational, and CSP, which relies on denotational semantics. This can lead to differences in how non-determinism is resolved, and can have an impact on security modelling. Roscoe et al pinpoint a weakness in BNDC [54], and demonstrate how it is corrected by SBNNI [54]. These properties were mentioned previously. In an earlier paper, Focardi [53] had shown that the lazy security of Roscoe et al [153], which was mentioned earlier, is equivalent to BNDC when applied to low-deterministic, non-divergent systems. In the language of process algebra, a system is said to be *divergent* if it can execute an arbitrary number of internal events.

Lowe [101] identifies flaws in the non-interference formulations of Ryan [158], Allen [11], Graham-Cumming [70] and Jacob [86], which are also due to different possibilities for non-deterministic choice in CSP. Lowe's solution is to annotate process transitions with additional syntactic information, and to introduce a "demon" that resolves such choices consistently. He observes that probabilistic models of CSP do not always solve the problem of how non-determinism is resolved. I discuss this further shortly.

#### 2.7.1.4 Function Views

In several works, function views of a system are used to describe information flow, as, for example, in Sutherland's non-deducibility [178] and McLean's selective interleaving functions [118]. More recent work includes the "function view" framework of Hughes et al [84] and *f-secrecy*, defined by Halpern et al [78], [77]. $f$-Secrecy has also been extended to model anonymity [79]. Another example is in opacity, which incorporates an observation function [27], [26], [25].

#### 2.7.1.5 Modal Logics

Modal logics have been found useful for reasoning about security. In the work of Cuppens [43], [41], [40], Cuppens et al, [42], [44], [43], [45], a combination of epistemic [81] and deontic [120] logic is used for an extensive analysis of security. Hughes et al's function view framework [84] and $f$-secrecy [77] are based on epistemic logic. Kripke [95] structures have also been used, for example in [178], [84].

#### 2.7.1.6 Synchronism vs Asynchronsim

Most of the existing work on information flow appears to be divided between asynchronous and synchronous models, and also infinite and finite traces. Both appear to be important as a property defined under one set of conditions can altered if cast in another. There is an interesting demonstration of this in [77], which is discussed later in the chapter.

#### 2.7.1.7 Language-based Security

Language-based security examines the problem of secure information flow at the program level. A detailed discussion is beyond the present scope, but is mentioned where relevant in the following sections. A good survey of work in this particular area can be found in [166].

### 2.7.2 Compositionality

As mentioned earlier, achieving compositionality was a motivation behind some of the research in secure information flow [110], [75], [89], [118]. It is well known that non-deterministic security properties, unlike safety and liveness properties, tend not to be compositional [118], [160]. Typically, safety properties assert that certain events must never occur, and liveness properties, that certain actions must eventually happen [97], [172], [12]. On the other hand, security properties are usually defined such that if certain behaviours occur, then there are other actions which should or should not take place. Lack of compositionality is attributed to the fact that security properties are generally defined over *sets of traces* rather than single traces [160], as in safety and liveness properties. Hence they tend to defy the Abadi-Lamport Composition Principle [9], which only holds for safety and liveness properties.

Nevertheless, compositionality is essential for verifying that systems built from secure components are secure. An important part of software engineering, for example, is modularity, that is, the ability to build systems from COTS products or to re-use library modules. Given the growing demand for secure software, it would seem desirable that the specified security properties for a system under construction should be compositional.

As an aside, it is interesting to note a further consequence to the difference between trace and non-trace properties. In [172], Schneider shows that safety properties can be enforced by execution monitoring, represented by an automaton, *EM*. However, *EM* can stop executions, but not predict or force them. Hence information flow and integrity properties cannot be enforced. Pursuing this theme, Ryan discusses the compromises that are often made by merging social with technical issues. An example is using access control, which is *EM*-enforceable, to implement safety requirements, and trusting users to observe integrity policies [161].

#### 2.7.2.1 Hook-up Security

As mentioned in Section 2.4.2, *hook-up security* [110] was motivated by the desire for a compositional security property. According to the original definition, the property holds if, for all traces of a system, $\alpha = \beta\gamma$, and sequence $\alpha'\beta'\gamma'$ formed by adding or deleting High inputs, there exists a trace $t''$ that differs from $t$ only in High outputs after the first change. However, McCullough found later that hook-up security was not compositional. In his later modification known as *restrictiveness* [112], High outputs can only be added after the final Low input in $t'$.

While it is compositional, a problem with restrictiveness, like hook-up security and non-deducibility, as mentioned earlier, is that it is only concerned with High inputs. McLean [115] also noted that restrictiveness may not hold in even functionally correct implementations of restrictive specifications. Refinement techniques tend to increase the predictability of a system by reducing non-determinism, hence a security property can fail in the refined system. This is also known as the *refinement paradox* [85], and can occur with any security property, not only restrictiveness. Ryan discusses various CSP approaches to solving this problem in [160], and proposes a definition for non-interference based on power bisimulation, which is preserved under refinement and composition. In [153] Roscoe et al propose a definition of non-interference, *lazy security*, in terms of a low view which remains deterministic: hence the property is preserved under refinement.

Johnson et al [89] were also interested in system composition, and, in an attempt to simplify hook-up security, proposed *correctability*, which is defined in terms of *perturbations* and *corrections* of traces. A perturbation of a trace, $t$, is a sequence, $t'$, by adding or deleting a High input. A correction is a trace, $t''$, formed from $t'$ by adding or deleting a High output. A system is ***high-event-correctible*** if for all traces, $t$, and perturbations of $t$, there is a correction. In addition, a perturbation must directly precede the final segment of a trace which has no High inputs. To achieve compositionality, a weaker property *forward correctability* was defined in [89], that is, every perturbation has a correction, but the perturbation can occur just before the final Low input in $t$.

Correctability and forward correctability both have a weakness in that only High inputs are of concern. Wittbold et al [186] compared forward correctability with their property, non-deducibility on strategies, and argued that the former was too strong. However, they did not investigate the compositionality aspects of non-deducibility on strategies.

## 2.7.3 Possibilistic vs Probabilistic Models

Many of the security properties discussed in previous sections are possibilistic, in that they are defined over possible executions of a system. Mclean [117], [115] and others have pointed out the limitations of such models, and argue that it is necessary to consider probabilities in designing secure systems. Suppose, for example, that a system satisfies non-interference. On observing any low-level event, $l$, Low should have no knowledge of any High events. However, if Low is 99% certain that a low-level event $l$ is caused by a High event, $h_1$, security can no longer be assured.

Nevertheless, it is simpler to reason with a possibilistic model, which could then be extended to take probability into account, as in for example, [78], [77], [96]. This is very much the view I have chosen for the present work.

Some authors hold the view that probabilistic uncertainty can also be achieved through encryption, as for example, in [186], [84], and so it is justifiable to model a possibilistic observer, for example, the Dolev-Yao attacker [50]. However, Pfitzmann and Waidner show that this is no longer justified for modern cryptographic systems [15], [143]. A further discussion of the cryptographic angle, is however, outside the present scope.

A realistic standpoint is taken by Lowe, who argues that probabilities are only useful when they can be assigned accurately. Sometimes, the way in which a system will resolve non-deterministic choice is known. Otherwise, it may be preferable to assume the "worst case" distribution [104]. This also has its uses when compensating for under-specification in design.

One of the first probabilistic models for security was McLean's *Flow Model (FM)* [117]. Based on a synchronous trace system, there are two functions, each giving the probability of a user's output and input at a given time. Gray formalised the FM by the requirement that Low output be independent of previous High events [72]. Building on this model, Gray et al define *probabilistic non-interference (PNI)* [73], which states that the probability of any Low event should not depend on any High event. Interestingly, there are intuitive similarities with the modelling of *strategies* in [186]. In fact, Halpern et al have shown that PNI is equivalent to their definition of $f_{strat} - S$-secrecy, which they also equate with non-deducibility on strategies [78]. In [160], Ryan's formulation of probabilistic non-interference states that probabilities in an evolving system are unchanged by High events.

Backes et al extend earlier work on transitive, probabilistic non-interference [16] to the intransitive case in [15]. Both models consider complexity aspects, while the later work draws on cryptographic notions, for example, error probabilities and refinement proofs through *simulatability*.

Related ideas are Lowe's use of demons to resolve non-deterministic choice [101], which was mentioned earlier in the chapter, and also his qualitative approach to intransitive information flow [101]. Similarly, Di Pierro et al's *approximate non-interference* [144], in which a system is secure if the chance of an attacker learning a secret is less than some constant $\epsilon$.

## 2.8 Anonymity

### 2.8.1 Background

There are numerous situations where anonymity might be desired; voting, donation and publication being just a few of them. As with any security property, the definition of anonymity can be elusive. An added problem is inference on the part of an attacker, and there can never be absolute certainty about how much knowledge the attacker has, or can acquire. This would seem to be the motivation behind $k$-anonymity [181], [180], where a system is anonymous up to a certain quantifier, $k$, for example, individuals. In other words, there must be at least $k$ identities which, if connected with some action or information, are indistinguishable to an attacker.

With the increasing use of the Internet, preserving anonymity has become even more of a challenge. Withholding identity during the course of remote business transactions and personal communication in a complex inter-connected environment, whilst maintaining the appropriate trust relationships, is a difficult task. As will be discussed shortly, this is typically achieved through the use of pseudonyms.

The term 'anonymity' is often used to mean different things. For example, it may be required that not only the identity of the sender, but also the recipient of a message should remain hidden. This is also sometimes referred to as privacy, although here, it is perhaps more precisely defined as relationship anonymity [84]. Other terms also occur in relation to identity hiding, such as "unlinkability" and "unobservability" [84]. Pfitzmann et al [140] have attempted to clear the confusion with a proposal for terminology, but this has by no means been accepted as standard. They describe unlinkability as the inability to connect two or more "items of interest" (IOIs) [140], such as messages, events or actions, and unobservability as indistinguishability of IOIs. Hughes et al refer to these various forms of anonymity in their case study of anonymity [84].

It would seem that networked communications have added complexity, not only to ensuring anonymity, but to the very definition of the term itself. As a result, one needs to be clear about what is being hidden, who it is being hidden from, and how well it must be hidden [79]. The "how well" aspect has manifested itself in various formalisations that include various strengths of anonymity: strong [176], probabilistic [79], [31], [137] and conditional [31], [137].

Forms of weak anonymity are also possible, such as pseudonymity, in which an entity performs actions under an assumed identity. Pseudonymity requires that such actions cannot be linked to the true identity, although it may be possible to connect them to the pseudonym. As an example, in Internet gaming, players are often only known to each other under the guise of a character in the game. Pseudonymity also appears in schemes which allow, for example, an entity to carry out remote commercial transactions under an assumed identity. In this case, the pseudonym may be something publicly verifiable, such as a digital signature, which is used to establish trust between the parties conducting business [32], [33].

Research in anonymity appears to be split between formalising the concept of anonymity, anonymity protocols, and methods for ensuring anonymity. The work of Reiter et al in [148], could perhaps be regarded as an exception, as it proposes both definitions for various levels of anonymity and an implementation. Their definitions are, however, informal. Some of their concepts, notably *probable innocence*, is formalised in later work by Palamidessi [137] and Palamidessi. et al [31], [22]. Formal methods which have been used for modelling anonymity range from CSP [176] to automata [137], [31], and there is also some work which incorporates epistemic logic [182], [84], [79].

The volume of work on implementations of anonymity far outweighs that of formal definitions. Nevertheless, the focus here will be on the latter, which is perhaps more directly relevant to this work.

However, for completeness, mention will also be made of protocols and implementation methods, particularly of mix-nets [34], which feature in many voting systems. Essentially, a mix-net is a way of passing data through a series of randomised nodes, in such a way as to destroy the link between original source and end-point of each data item. Care must be taken to ensure that the system cannot be corrupted, for example, by a node deleting or replacing data passed thorough it. One way of doing this is by auditing the mix process [35], [36], [87]. In some voting schemes [35], [36], [132], [133], mix-nets are a way of anonymising votes, making it difficult to link the encrypted receipts displayed on a Web bulletin board with the final decrypted votes. Mix-nets are discussed in later chapters of the thesis.

In this section, the various ways in which anonymity has been formalised are discussed first. This is followed by an overview of methods for ensuring anonymity and pseudonymity, and the different situations in which they may be applied. A summary and discussion concludes.

## 2.8.2 Formalisations

### 2.8.2.1 Strong Anonymity

One of the earliest formal definitions of anonymity, and one often cited in the literature, is by Schneider et al [176]. Based on the process algebra CSP [83], they define strong anonymity in terms of a process, $P$, and an anonymity set, $A$, that is, the events to remain hidden. Using this notation, $P$ is strongly anonymous if it makes all events from $A$ possible whenever any one of them is. It can never be certain which event from $A$ occurred, since they are all equally likely. Another way of expressing this is that "every anonymous event is independent of every other such event" [176]. The definition is verified in an analysis of the Dining Cryptographers protocol [34]. The simplest example of the protocol is where two cryptographers are dining out, and by a clever arrangement of tossing coins, they can find out whether either one of them paid the bill or the National Security Agency did. If either of the cryptographers paid, to an outside observer, the identity of the payer will still be anonymous, but the level of anonymity will be weaker than that defined above. In variations of the protocol with more than two cryptographers, the identity of the payer can remain anonymous, which corresponds to strong anonymity.

Schneider et al also discuss the possibility that a weaker form of anonymity than the one proposed may sometimes be more appropriate. An example is a voting system, where it may only be necessary to hide the identity of a voter, and not the value of a vote. It would be sufficient to ensure that two votes, such as $a1.v1, a2.v2$ are generated by different agents. For strong anonymity, the trace $\langle a3.v1, a3.v2\rangle$ would have to be possible, in which case the same agent, $a3$, could have voted either $v1$ or $v2$.

A definition of anonymity as opacity is a subject of future research. Halpern et al [79] have also compared their definition of anonymity, which is based on their secrecy property [77], to strong anonymity. This work will be discussed in further detail shortly.

#### 2.8.2.2 Formalisation of Anonymity using Group Principals

In [182], Syverson et al have provided not so much a definition, but an axiomisation for different forms anonymity, based on the notion of group principals and epistemic logic. Essentially, a group principal has a set of members, and its properties depend on its type under the standard mathematical operators. For example, an or-group possesses knowledge under disjunction, that is, it knows at least what one member knows. However, the formulae used in the axioms for anonymity, for example, $\varphi P$, are not specific to type of principal, $P$. The formulae generalise actions such as "$P$ said to $Q$". An example of one such axiom is "possible anonymity", which describes the situation where an attacker cannot eliminate the possibility that some formula is true.

Syverson et al argue that their characterisation is more expressive than that of Schneider et al [176], allowing for variations in principal, formulae and attacker, and hence, in different anonymity properties. They illustrate their approach by analysing a Web proxy service for anonimising messages [13]. However, the lengthy and complex manual analysis involved would tend to give methods with automated tools an advantage. In [176], for example, Schneider et al use the CSP model-checker, FDR [51]. For further details and the complete set of axioms, the reader is referred to [182].

#### 2.8.2.3 Anonymity in the Function View Framework

Hughes et al [84] also propose an extensive set of anonymity properties as a case study for their function view framework. Function views, as discussed in the first part of this chapter, represent partial knowledge of a system. In their model, information hiding has been defined as an observational equivalence over system configurations, which are "possible worlds". Attributes and sets of functions, each for some particular configuration, are used to model communication. An example is the sender function $s : M \rightarrow A$, where $M$ is an edge of a protocol graph, and $A$, a vertex. Edges of a graph represent abstract conversations, and are coloured to represent relationships between the associated vertices, which represent agents. Views on functions, such as the sender function above, can model various levels of anonymity. An example is $k$-value opaqueness, which is equivalent to $k$-anonymity [181]. This property stipulates that from the attacker's viewpoint, there are $k$ possible senders of the message $M$. The full set of anonymity properties is found in [84].

#### 2.8.2.4 Anonymity in the Secrecy Framework

Extension of the framework to model probabilistic properties is briefly discussed in [84], but Halpern et al later argue that to do so would not be straightforward [79]. They show that their expression anonymity as a variant of the *secrecy* property [79] can be adapted to define possibilistic anonymity.

Recall that, intuitively, $\varphi$ is secret if an agent, $A$ never knows the truth of $\varphi$, that is, $A$ cannot eliminate the possibility of $\neg\varphi$. Secrecy is equated with non-interference, that is, the low-level users never learn anything about high-level users. By altering what needs to be hidden about the system, that is, the connection between identity and action, Halpern et al arrive at a definition for anonymity. *Minimal* and *total anonymity* are defined, and later compared with definitions of anonymity in other work. For example, total anonymity is shown to be equivalent to *strong anonymity* [176]. *Conditional anonymity* describes the situation where an agent never learns anything new about the system, that is, additional to what he knew before interacting with it.

### 2.8.2.5 Crowds

In [148], Reiter et al provide informal definitions of anonymity, which they demonstrate with *Crowds*, a probabilistic anonymous messaging service. Their definition of *beyond suspicion* would seem to coincide with total and strong sender anonymity [79], that is, a sender is no more likely to have sent a message than any other suspect. A weaker definition is *probable innocence*, which applies when a sender is no more likely to have sent a message than not to have sent it. A sender is *possibly innocent* if, from the attacker's viewpoint, there is a non-trivial probability that the message was sent by someone else. These definitions fall in the middle range of a scale, which runs from *absolute privacy* to *exposed*. The definitions can also be applied to the receiver, or both sender and receiver of a message.

The basic idea behind Crowds is that users are represented in a *crowd* by a local process called a *jondo*. When a member starts the jondo, it sends a request to a server, called the *blender*, for admittance to the crowd. If successful, the jondo, acting on the user's behalf, forwards messages to another randomly chosen jondo. This jondo flips a biased coin to determine whether or not to forward the request to another jondo. If successful, assigned a probability of $p_f$, it forwards the request, otherwise passes it to the end server. The process is repeated until the message reaches its destination, and a reply is returned via the same path. Likewise, subsequent requests.

Crowds is assessed for three different types of attacker: local eavesdropper, collaborating members and the end server. The level of anonymity provided for either the sender or receiver in the presence of an attacker, is given by number of formulae based on the size of the Crowd, $n$, $p_f$, and the number of collaborating members, $c$, As one might expect, assurance of anonymity increases with the size of a crowd. Further details can be found in [148].

Garcia et al [61] provide a definition for anonymity based on observational equivalence. However, they take a cryptographic approach, combined with epistemic logic. As cryptographic methods are beyond the scope of this thesis, it is not discussed further here.

#### 2.8.2.6 Probabilistic Anonymity

In common with other research on information flow security, there is a division of opinion on the advantages of probabilistic, as opposed to possibilistic, systems. The claim that incorporating probability gives stronger definitions [79], [137], [31] is countered with the belief that it can be misleading [182]. Certainly, one has to be able to assign probabilities accurately.

Palamidessi [137] joins the probabilistic camp with definitions for anonymity based on probabilistic automata. Essentially, the automata is a labelled transition system. For each node, the outgoing transitions are partitioned in steps: each step represents a probabilistic choice, while the choice between steps is non-deterministic. Non-determinism is modelled by the action of a scheduler. The automata can also be fully probabilistic. The aim of this work appears to capturing probabilistic protocols, such as Crowds [148], with either probabilistic or nondeterministic users. She presents definitions for both probabilistic and non-deterministic anonymity. Intuitively, a system is strongly anonymous if, given two schedulers, $\sigma$ and $\vartheta$, it is not possible to detect which scheduler has been chosen. Strong probabilistic anonymity is based on conditional anonymity, that is, $i$ is anonymous if observations do not increase the probability that $i$ performed some action, even if probabilities are known beforehand. Both non-deterministic and probabilistic weak anonymity are also defined. The basic idea is that the likelihood, and respectively, probability, of $i$ having performed some action, $a$, increases by an additive factor after $a$ is observed.

Following this work, Chatzikolakis et al [31] provide new definitions for probable innocence, as described in [148]. They claim that Reiter et al's definition has been misinterpreted as being dependent on probability of the users, when in fact it is not. This being the case, it would seem natural that a system such as in [137] would be well suited to a formal definition of probable innocence. Further details can be found in [137] and [31].

### 2.8.3 Pseudonymity

An example of formal work on pseudonymity is in the function view framework of Hughes et al [84]. Their anonymity case study is extended by defining additional functions which map pseudonyms to, for example, agents and messages. Building on their various definitions of anonymity, they discuss possibilities for modelling a class of pseudonymity properties.

The main body of work in this area appears to consist of methods for registering and administering pseudonyms in applications such as electronic mail servers. Examples are found in [171], [109], [183], [32]. In [33], Chaum proposes a scheme whereby an individual can carry out business transactions such as electronic payment under a digital pseudonym.

### 2.8.4 Implementations

So far, most of the research on anonymity appears concentrate on implementation rather than formal definitions. Pseudonym servers have been mentioned above.

There are numerous examples of systems for anonymising communication, which appear to fall broadly into three categories.

#### 2.8.4.1 Proxies

A *proxy* can be interposed between the origin and endpoint of a message, hiding the identity of the sender from the receiver, as for example in [7]. However, the problem with this is that the proxy acts a single point of failure. In addition, it is vulnerable to a passive attacker if the attacker can get control of the proxy [148].

#### 2.8.4.2 Mixes

Another method for ensuring anonymous communication is by using mixes [32], in which packets of equal length are sent through a series of mix-servers. Each server shuffles the data and forwards it to the next server. Typically, the data is altered cryptographically. For example, the data may be encrypted with the public keys of each of the mix-servers in reverse order, and then decrypted with the corresponding private keys at intermediate points along the route. Re-encryption nets are also possible, as, for example in [69].

Mixing achieves unlinkability between the two endpoints of a message. As mentioned earlier, this has been used in several voting systems [36], [35], [131], [132] as a way of ensuring that no links can be made between the encrypted ballot receipts and the decrypted votes. To maintain integrity, mixing can be done so that it is cryptographically verifiable, as, for example in [87], [133], [36]. This will be discussed in more detail in later chapters of the thesis.

Examples of other applications where mixes have been applied are electronic mail [74] and telecommunications services [141]. As mixes generally utilise public key encryption, they can be vulnerable to cryptographic attacks [48], [142]. Another problem is that the security of mixes can be compromised by collusion between mix-servers. Usually, at least one honest server must be honest to ensure unlinkability [35]. They have also been criticised as being costly due to successive decryptions or encryptions, or both [148]. However, it should be noted that encryption does provide security against a global eavesdropper [148].

#### 2.8.4.3 Onion Routing

The third method by which communication can be anonymised is by onion routing [147]. In this method, messages are sent via a series of proxies, with layered encryption containing the necessary routing information in reverse order. Each proxy strips off its corresponding layer. and passes the data on to the next proxy until the final destination is reached. The method of encryption is similar to mixes, but with mixing, data is shuffled in batches, whereas data is sent and received through the onion network on demand.

There are systems which fall between the three categories mentioned above. An example is Crowds [148], described previously, which appears to resemble onion routing, but does not utilise encryption. The intention is to optimise performance [148], but does so possibly at the expense of data confidentiality. It is assumed that in a multi- domain system, a global eavesdropper is unlikely. However, it is not clear whether or not each jondo has the capability of reading and recording information as it passes through. Another example is the successive decryptions performed as an encrypted vote is passed through an anonymising mix, such as in Prêt à Voter [36], to be discussed in more detail later.

Work has also been done on protocols for anonymising communication. In [123], for example, Molina-Jiminèz et al propose a scheme in which users can send messages from a PDA using transient, dynamically-assigned, random IP and MAC addresses. The closest analogy to this might be a non-personal, temporary pseudonym. Marshall at al [107] have also devised a system which allows individuals to communicate under pseudonyms, but permits them to be linked to their true identities should the necessity arise. Their protocol, however, was later found to be vulnerable to attack [126].

## 2.9 Discussion and Summary

Existing definitions for information flow security range widely, encompassing numerous different formalisms and at different levels of abstraction. Each of the security properties also take different operational aspects of a system into account and to varying degrees. This is usually the events to be allowed or disallowed, but can also include other factors such as time and probability.

There appears to have been, and continues to be, much debate in the field. A consensus has not, as yet, been reached on a universally correct definition. Aside from the difficulties caused by the complexity of computer systems, security requirements also differ. A model that provides security assurance in one system may fail in another. This was exemplified by "System Z" [116], which conformed to the BLP model but was obviously insecure.

Aspects such as compositionality could also be affected by system specification and design. Hence, a security property that is compositional in theory, may fail for the specified construct.

Systems may also differ in the types of security properties that are important, hence the need for properties such as non-leakage [185], which prevent deductions about initially secret information. Similarly, the various definitions of "weakened" non-interference, for example, [127], [144], are useful where total absence of information flow is not necessary or is impossible, or both. As a result of shifting trends in computing, certain properties may gain prominence. An example is anonymity, with the increasing number of Internet transactions.

It is perhaps unsurprising that complexity in networked communications has lead to a need for precision about security requirements for information. This has been reflected in the large number of anonymity-related terms that are used in the literature [140]. This being the case, system designers must be clear about the information that is to be hidden [79]. The information in question could be identity, or it could be both identity the link between a particular action and that identity. It could also be two or more identities involved in some transaction.

Some existing formalisations of anonymity and pseudonymity have been discussed, along with informal notions for related properties. There has also been a brief overview of existing systems and protocols, which have been proposed to ensure anonymity in various activities, such as messaging, voting and business transactions.

In some of these schemes, there is the danger of exposure from traffic analysis, as discussed, for example, in [87], [148], [123]. There is also a body of research devoted to traffic analysis attacks and proposed defences against them. For example, in mix-nets for voting schemes, a possible counter-measure is to inject dummy votes into the system before the mix process so that it is more difficult to trace a path from any one encrypted vote to the final decrypted value. Mix-nets will be re-visited later in the thesis.

It would seem that there can be no single definition of security and that assessment should be made on a per application basis [116]. The required properties could then be modifications or adaptations of pre-defined models. It may be preferable, to regard existing definitions as guidelines rather than as arbiters of security.

# Chapter 3

# Opacity

In the previous chapter, I reviewed some of the many existing information flow properties, mainly concentrating on the well-known properties, such as non-interference and non-deducibility. These are often used as a starting point for new definitions of security, or as comparisons for more recent work. However, I also discussed a few properties that have emerged recently, as they appear to be leading the field in new and interesting directions. One example is secrecy [78], [79], which seems to adapt readily to various notions of secure information flow, such as, non-interference and anonymity, as well as modelling probability.

Another relatively new and interesting property is opacity, the study of which forms the basis of my thesis. Intuitively, a predicate, $\phi$, is opaque if it is impossible for an observer to establish the truth of $\phi$. For a system described by a labelled transition system, $\Pi$, if $\phi$ is true for any run of $\Pi$, then there exists an observationally equivalent run where $\phi$ is false [25]. The predicate, $\phi$, can be any property of interest and the notion of equivalence can be varied, so opacity remains flexible to the requirements of the system under consideration. As will be demonstrated in the ensuing work, this flexibility gives opacity the ability to capture various notions of secure information flow.

The notion of opacity may have originated from the work of Hughes et al [84]. In their framework, partial knowledge of a system is expressed as various forms of "opaqueness" under different function views, which are mathematical abstractions for partial knowledge of a function. This work has been described in the previous chapter. Subsequently, in the analysis of security protocols, Mazaré used the term "opacity" to describe the inability of an observer to establish the truth of a predicate over system traces [108]. Although both ideas are strongly related, the second is perhaps closest to the present work. Opacity, in Mazaré's sense of the word, has been given a rigorous, formal treatment by Bryans et al [25], [26], [27]. Initially, they use Petri nets [26], [27] as a modelling tool, but they later generalise the framework using labelled transition systems [25].

Referring to the informal definition above, opacity is expressed in terms of an observational equivalence. Hughes et al [84] show that "opaqueness", in their framework, lifts to a predicate over observational equivalence. by extension, opacity could be said to describe partial knowledge of the system.

Recall that in [77], [78], Halpern et al propose the secrecy property, which may appear to resemble opacity. Intuitively, a formula $\varphi$ is secret if an observer never knows that $\varphi$ is true. There are, however, differences in the way the properties are expressed. Secrecy is based on epistemic logic, and is not expressed in terms of an observational equivalence, but in terms of an agent's knowledge.

It should be noted that "observational equivalence" may have an overloaded meaning. In the semantics of process algebra, two processes are observationally equivalent if they have the same behaviour, taking internal events and hence, non-determinism, into account [83]. The processes could also be regarded as bisimilar, or indistinguishable. However, in expressions of opacity [25], [26], [27], and apparently also in the work of Hughes et al [84] and Halpern et al [77], [79], [78], observational equivalence is used in the sense of trace equivalence, that is, the two processes share the same set of observable behaviours. Whether or not this provides a sufficiently robust model is a decision for the system designer. I return to this issue later in the chapter.

Lakhnech et al [96] have formulated probabilistic opacity for verifying cryptographic protocols, that is, "even if the adversary is 99% sure of the truth of the predicate, it remains opaque as the adversary cannot conclude for sure" [96]. They claim that it can be applied to "classical" opacity [25] in a straightforward way. For simplicity, the work in this thesis is based on a possibilistic system, with a view to possible extension in the future. Although probabilistic opacity is not discussed further here, it would be interesting to investigate in the future.

In this chapter, formal definitions for the opacity property and its variations will be given in the labelled transition system style of Bryans et al [25], [26], [27]. Following this, variations of opacity are mapped to several well-known security properties [139]. I then review some related work, before a discussion and summary.

## 3.1 Opacity: Formal Definitions

I now present a formalisation of the opacity property [25], along with some related notions that will be used in later parts of the thesis. Full details can be found in [25]. As mentioned above, the opacity framework is currently based on the labelled transition system, which is defined below [25].

**Definition 1** *A labelled transition system (LTS) is a tuple* $\Pi = (S, S_0, L, \Delta)$*, where* $S$ *is the (potentially infinite) set of states,* $L$*, is the (potentially infinite) set of labels,* $\Delta \subseteq S \times L \times S$*, is the transition relation, and* $S_0$ *is the set of initial states. We consider the underlying LTS to be deterministic and so for transitions* $(s, l, s'), (s, l, s'') \in \Delta$ *it is the case that* $s' = s''$.

This provides a very general model, which can be adapted for other formal methods. It was found to be more convenient, for example, to use the process algebra CSP [83] for some of the work later in this chapter. Note that a non-deterministic system can be modelled as an abstraction of the underlying deterministic LTS.

A run of the LTS can be defined as follows [25].

**Definition 2** *A run of the LTS is a pair $(s_0, \lambda)$, where $s_0 \in S_0$, $\lambda = l_1 \dots l_n$, $l_i \in L$ is a finite sequence of labels such that there are states $s_1, \dots, s_n$ satisfying $(s_{i-1}, l_i, s_i) \in \Delta$ for $i = 1, \dots, n$. We will denote the state $s_n$ by $s_0 \oplus \lambda$, and call it reachable from s. The set of all runs is denoted by run($\Pi$).*

In the rest of this section, let the LTS be $\Pi = (S, S_0, L, \Delta, )$, as defined above, and $\Theta$, a set of elements called *observables.* Next, are definitions for possible variations of an observation function, *obs*, which can be used to model the ability of observer to gain information about the system [25].

**Definition 3** *Any function obs: run($\Pi$) $\rightarrow \Theta^*$ is an observation function. It is called label-based and: static/dynamic if respectively the following hold (below $\lambda = l_1 \dots l_n$):*

- *static: there is a mapping obs' : $L \rightarrow \Theta \cup \langle\rangle$, where $\langle\rangle$ is the empty trace given by $t \frown \langle\rangle \equiv t$, and such that for every run $(s, \lambda)$ of $\Pi$, $obs(s, \lambda) = obs(l_1) \dots obs'(l_n)$.*
- *dynamic: there is a mapping obs' : $L \times L^* \rightarrow \Theta \cup \langle\rangle$ such that for every run $(s, \lambda)$ of $\Pi$, $obs(s, \lambda) = obs'(l_1, \langle\rangle) obs'(l_2, l_1) \dots obs'(l_n, l_1 \dots l_{n-1})$.*

Static functions only allow the observer to interpret the same label in the same way. Dynamic functions model an observer with a potentially infinite memory to store labels, but can only use knowledge of previous labels to interpret a label [25].

Orwellian and m-Orwellian functions have also been defined, but are not used in this work. They are described informally later in this section. For full details, the reader is referred to [25].

The observation function can also be state-based, an example of which is defined as follows [25].

**Definition 4** *A state-based, static observation function is one for which there is obs' : $S \rightarrow \Theta \cup \langle\rangle$ such that for every run $(s, l_1 \dots l_n)$, we have $obs(s, l_1 \dots l_n) = obs'(s) obs'(s \oplus l_1) \dots obs'(s \oplus l_1 \dots l_n)$.*

In the present work, I only refer to the static, label-based *obs* function. However, there are possible extensions of opacity where dynamic or state-based functions, or both combined, may be more appropriate. These will be discussed in later parts of the thesis.

Returning now to formalising opacity, it may be desirable to determine whether an observer can establish the truth of $\phi$, a predicate over system states and traces, only from the result of an observation function, *obs*. Here, the concern is whether the observer can deduce that an observed execution belongs to the set of runs for which $\phi$ holds. This leads us to a general definition for opacity [25], which can be varied according to the requirements of the particular application:

**Definition 5** *A predicate $\phi$ over run($\Pi$) is opaque w.r.t. the observation function obs if, for every run $(s, \lambda) \in \phi$, there is a run $(s', \lambda') \notin \phi$ such that $obs(s, \lambda) = obs(s', \lambda')$.*

Initial-, final- and total-opacity have also been defined in [25]. Other variations of opacity. could also be obtained by altering the *obs* function. However, these are peripheral to the present work, and will not be pursued further here.

It is worth noting that opacity is an asymmetric property, that is, from an observed execution of the system, we are only interested in whether an observer can establish that the underlying run belongs to $\phi$. If, instead, we were interested in establishing whether the underlying run does not belong to $\phi$, we would consider instead the property $\overline{\phi} = run(LTS) \setminus \phi$ [25].

In [79] Halpern et al discuss the symmetry versus asymmetry of properties, but in a different sense to that used above. They observe that total secrecy is a symmetric property in that no information should flow from High to Low levels, but as a consequence of the definition, information does not flow in the opposite direction either. This can sometimes be too strong a requirement, as information flow from Low to High levels is not usually considered a breach of security. $f$-Secrecy, however, shares the asymmetric nature of opacity, that is, it is specified with respect to the knowledge of a certain agent. Similarly, opacity is characterised by observational equivalence with respect to, for example, a Low-level user, and does not account for High-level observation of the system. However, in a later section, I investigate whether or not there is a relationship between "symmetric opacity" and non-interference [139].

As noted earlier, general opacity is based on observational equivalence, which is similar to trace equivalence in the semantics of process algebras. Under certain circumstances, this may be too weak. Discussions on the limitations can be found in, for example, [160], [56], [154]. The trace model emulates a "passive observer", that is, one who is only able to observe a certain set of events, or labels. The adversary may, however, be able deduce certain (disallowed) information from system responses to certain (permitted) events, and a stronger model such as (stable) failures [83] may be necessary. In the opacity framework, this could possibly be achieved using a state and label-based *obs* function. With access to certain information about system state as well as labels, the adversary may be in a stronger position than one who is only able to observe events. This would be a useful extension to the current framework, and an interesting subject for future work.

As previously mentioned, opacity is a parametric property in which the predicate, $\phi$, can be specified according to the particular system or application under consideration. $\phi$ could, for example, be defined as being true of a trace if it contains High-level events. This has been done in many the definitions to follow.

## 3.2 Security Properties Cast as Opacity

One of the advantages of the opacity framework is its generality, in that the predicate, $\phi$, can be defined according to the security requirements of the system. Likewise, the observation function can be varied to reflect the power of the observer.

In this section, possible relationships between variations of opacity and other existing security properties are examined [139]. The properties investigated are non-interference [66], non-deducibility [178], non-deducibility on strategies [186], non-inference [134], non-leakage [185] and non-influence [185]. The aim is to test the flexibility of the opacity property further, and to learn more about its characteristics in relation to existing definitions of secure information flow.

As will be seen shortly, there does not appear to be a direct mapping between non-interference and opacity. On the other hand, both non-deducibility and (a special case of) non-leakage can, in each case, be cast as variations of the opacity property. The predicate, $\phi$, is, in some cases, defined as being true of a trace, $t$, with respect to the *obs* function available to a Low-level observer, if $t$ contains High-level events. The exceptions are with non-deducibility and non-leakage, where other variations of opacity are defined. The details are given in the relevant sections. I begin by discussing the possible link between opacity and non-interference [139].

### 3.2.1 Non-interference as Opacity

Non-interference, first proposed by Goguen et al in [66], was discussed in Chapter 2. Intuitively, no High commands should affect what observations Low can make of the system.

The original form of non-interference is, however, limited to deterministic systems, which can be problematic in practice. Non-determinism introduces an additional complexity which makes this a rather difficult task, but there have been numerous proposals for non-deterministic non-interference. Examples are recent definitions based on process algebras [164], [160], [55]. It is feasible, however, that any subsequent re-interpretation may subtly alter a security property, possibly even change it completely. For this work, the original definitions of the various properties are followed as closely as possible, rather than any of the subsequent re-statements by different authors. Hence, in this section, I refer to deterministic non-interference as defined by Goguen et al [66],[67] and given earlier in Chapter 2.

It may appear, at first, that non-interference can be cast as opacity, with the predicate, $\phi$, defined as being true of a trace if it contains High events. Translating the intuitive definition into opacity terms, for all runs of a system, there is an observationally equivalent run without any High events. However, a process which satisfies opacity with respect to $\phi$ may not, in fact, be non-interfering. This is illustrated with a simple process, $P$, in Figure 3.1 [139]. $P$ satisfies opacity, since, for every trace with a High event, there is a corresponding trace without. However, $P$ does not satisfy

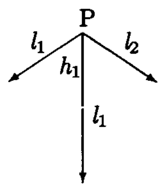

Figure 3.1: A process $P$ which satisfies opacity

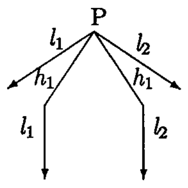

Figure 3.2: A process $P$ which satisfies non-interference

non-interference. On observing the $l_2$, for example, Low can eliminate $h_1$.

To satisfy non-interference, $P$ could be modified as shown in Figure 3.2 [139]. Low is now prevented from ruling out any High event, and so High does not interfere with Low. At the same time, opacity still holds for $P$. This suggests that non-interference is a stronger property than opacity, and that there is a relationship by implication, rather than a straight mapping between the two properties. More precisely, non-interference implies opacity, but the converse is not true [139].

#### 3.2.1.1 Proof

There now follows a proof that non-interference implies a suitably defined form of opacity, but that the converse is not true [139].

First, the *purge closure property* (PCP) is defined. This states that for all traces of the system, $\tau S$, the purge of a trace is a valid trace of $S$, where "purge" means that all High level events have been removed. In other words, for any trace with High-level events, there is a valid trace with the same sequence of Low level events, but no High events. A process P satisfies the purge closure property if

$$\forall\, t \in \tau P, purge_H(t) \in \tau P$$

Next, a few CSP notions are introduced. Full details on CSP can be found in [83] and a general introduction in [160]. The interface to a process $P$ is represented by its alphabet, the set of externally visible events through which it interacts with the environment. For deterministic processes, the set of events that it will offer the environment after a given trace will be well-defined. Since only deterministic processes are considered here, the *initials* can be used, rather than refusal or acceptance sets. The *initials* of $P$ are the events that $P$ is prepared to participate in next, defined as

$$initials(P) = \{a \mid \langle a \rangle \in traces(P)\}$$

For notational convenience, the initials are restricted to a subset of the alphabet, for example, the set of Low-level events, $L$. $Init_L$ are the initials restricted to $L$.

The non-interference of Goguen et al can now be cast in the following way [139].

A deterministic process S is non-interfering if

$$\forall\, t \in \tau S, Init_L(S/t) = Init_L(S/purge_H\, t)$$

where $S/t$ is the process $S$ after trace $t$.

Strictly speaking, this is more general than the original definition, as Goguen et al simply required that next, low outputs be equal and assumed that these were unique for a given input sequence.

The proof that non-interference (NI) implies a suitably defined form of opacity can be summarised as follows [139]:

$PCP \Leftarrow NI$ and $opacity(\phi) \Leftrightarrow PCP$, but $PCP \not\Rightarrow NI$. Hence $NI \Rightarrow opacity(\phi)$ but $opacity(\phi) \not\Rightarrow NI$.

$opacity(\phi)$ is defined shortly.

The proof can be split into two main steps. The first step is to show that non-interference implies the PCP, that is,

$$NI(S) \Rightarrow t \in \tau S \Rightarrow purge_H(t) \in \tau S$$

.

*Proof.* $S$ is also shown to satisfy the PCP by contradiction. Suppose $S$ satisfies non-interference but not the PCP. There must then exist a smallest trace of $S$ that violates the PCP, since traces form a well-ordered set. It is convenient to represent this smallest PCP-offending trace as $t \frown \langle e \rangle$, i.e. the trace $t$ followed by a single event, $e$, with $\langle e \rangle \in L$.

$t \frown \langle e \rangle \in \tau S \wedge\ e \in L$ but $purge_H(t \frown \langle e \rangle) \notin \tau S$, since $e \in L$, $purge_H(t \frown \langle e \rangle) = purge_H(t) \frown \langle e \rangle$.

Since it was assumed that $t \frown \langle e \rangle$ was minimal, then $purge_H(t) \in \tau S$.

So $e \notin Init_L(purge_H(t))$, but $e \in Init_L(t)$, which contradicts the assumption that $S$ satisfies non-interference.

□

In the second step of the proof, the PCP is shown to imply a suitably defined opacity property and vice versa, that is, $PCP \Leftrightarrow opacity(\phi)$.

The predicate $\phi$ is first defined as follows:

$opacity(\phi)$, where $\phi(t) = T$ if and only if $t \upharpoonright H \neq \langle\rangle$

In words, if $\phi$ is true for $t$, then $t$ has at least one event from the set of High events, $H$.

$S$ satisfies opacity with respect to $\phi$, defined as above, and the appropriate obs function if $\forall\, t \in S,\ if\ \phi(t) = T, \exists\, t' \in \tau S$ such that $(t \upharpoonright L = t' \upharpoonright L) \wedge \phi(t') = F$.

This step of the proof is split into a further two steps. The first step is to prove $PCP \Rightarrow opacity(\phi)$, that is, the PCP implies opacity($\phi$).

*Proof.* If $\phi(t') = F$, then $t'$ has no High-level events, which implies $purge_H(t') = t'$. The PCP tells us that $\forall\, t \in \tau S$, $purge_H(t)$ is also a valid trace of $S$. Clearly, $\forall\, t \in \tau S \cdot \phi(purge_H(t)) = F$. So $PCP \Rightarrow opacity(\phi)$.

□

The second step is to prove that $opacity(\phi) \Rightarrow PCP$.

*Proof.* From the above, if opacity holds, then $\forall\, t\ \exists\, t' \in \tau S$ s.t. $\phi(t') = F \wedge t \upharpoonright L = t' \upharpoonright L$.

□

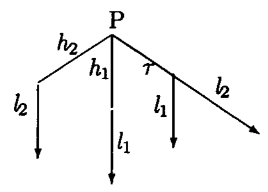

Figure 3.3: A process $P$ which satisfies symmetric opacity but not non-interference

The next step is to prove that $PCP \not\Rightarrow NI$.

*Proof.* This is done by a counter-example. I refer the reader to the process $P$ in Figure 3.1. $P$ satisfies the PCP but not non-interference. Observing $l_2$ proves $h_1$ did not occur. Note that, as shown in Figure 3.2, if the Low-level user observes either $l_1$ or $l_2$ it is possible that $h_1$ did not occur.

□

It has now been established that $NI \Rightarrow PCP$ and that $PCP \Leftrightarrow opacity(\phi)$.

Therefore, the final step of the proof, $NI \Rightarrow opacity(\phi)$ follows by transitivity [139].

□

#### 3.2.1.2 Symmetric Opacity

As mentioned earlier, opacity is an asymmetric property in that the concern is only whether the observer is able to establish $\phi$, or alternatively, $\overline{\phi} = run(LTS) \setminus \phi$.

Pursuing the study of a relationship between opacity and non-interference, a variation of opacity is considered, symmetric opacity, in which the observer should not be able to establish either $\phi$ or $\overline{\phi}$ [139].

**Definition 6** *A system ($\Pi$) satisfies symmetric opacity with respect to $\phi$ and the observation function obs if, for every run $(s, \lambda)$, both $\phi$ and $\overline{\phi}$ are opaque.*

Intuitively, the observer should not be able to establish either $\phi$ or $\overline{\phi}$. With the predicate, $\phi$, defined as being true of a trace if it contains High-level events, the Low-level observer should not be able to establish whether or not a High-level event has occurred. This suggests that there may be a relationship between non-interference and symmetric opacity. However, Figure 3.3 shows that this is not the case. $P$ satisfies symmetric opacity with respect to both $\phi$ and $\overline{\phi}$, but not non-interference.

### 3.2.2 Non-deducibility

One of the first definitions for secure non-deterministic information flow was proposed by Sutherland in [178]. Referring to his "possible worlds model", given in the previous chapter, a system satisfies non-deducibility if there is always a world, $W''$ that is compatible with both $W$, viewed by a Low user and $W'$, in which a High user makes some input. Intuitively, no deductions about High-level inputs should be made from any Low-level observation.

More recent expressions of non-deducibility involve traces of the system, for example, in [160], [42]. Intuitively, for any two traces, $t, t'$, there is a trace, $t''$ which contains the same High-level inputs as $t'$, and gives the same Low-level observations as $t$.

As discussed in Chapter 2, a problem with non-deducibility is that the only concern is High-level inputs. Sutherland reasoned that nothing significant could be learned from High-level outputs if they are not connected with High-level inputs. Guttman et al later showed this to be false, and proposed a modification of non-deducibility, which they called non-deducibility on outputs [75]. Informally stated, any Low-level observation should be compatible with any possible High-level inputs and outputs. This is essentially a restatement of non-deducibility, but is concerned with both High inputs and outputs.

As shown in the following, non-deducibility on outputs can be cast as opacity, with $H$ denoting the set of High inputs and outputs [139].

First, a predicate, $\phi_t$, is defined as

$$\phi_t(t') := t \upharpoonright H \neq t' \upharpoonright H$$

.

That is, $\phi_t$ is true of a trace $t'$ when the High-level projection of $t'$ is not equal to the High projection of $t$.

The static *obs* function $obs_L$ is defined as

$obs_L(e) = e$ if $e \in L$
$obs_L(e) = \langle\rangle$ if $e \in H$, where $\alpha(S)$ is partitioned between L and H.

$S$ is defined as totally High opaque with respect to $obs_L$ if and only if

$\forall\, t \in \tau(S) \cdot \phi_t$ is opaque.

Expanded, this is equivalent to

$$\forall\, t, t' \in \tau(S),\ \exists\, t'' \text{ such that}$$
$$\phi_t(t'') = \mathrm{F} \wedge obs_L(t) = obs_L(t')$$

or

$$\forall\, t, t' \in \tau(S),\ \exists\, t'' \text{ such that}$$
$$(t \upharpoonright H = t'' \upharpoonright H) \wedge (t' \upharpoonright L = t'' \upharpoonright L)$$

which matches the definition given in Section 2.4.2.

□

Another problem with non-deducibility, as demonstrated in [186], is that it allows covert channels. It appears that non-deducibility on outputs also suffers from this vulnerability. Wittbold et al proposed a further modification which they called non-deducibility on strategies, where a strategy for a process, either High or Low-level, determines an input from a given sequence of previous inputs and outputs. Intuitively, any Low-level observation should be consistent with any High-level strategy, and for consistency to hold, there must be a trace that is compatible with both.

It is possible that non-deducibility on strategies can be cast as opacity using the formulation above, but with $H$ denoting a High strategy. This would be interesting to investigate in the future.

## 3.2.3 Non-inference as Opacity

Another proposal for a non-deterministic security property is *non-inference* [134], which was discussed previously in Chapter 2. O'Halloran's definition, also given in Chapter 2, implies that for any trace of the system, there is either a corresponding trace without High actions, or Low does not observe any activity. Low cannot therefore eliminate the possibility that High did nothing. As noted by McLean [118], non-determinism is captured by the possibility of High output without High input.

The original definition was based on CSP, in which by prefix closure, the empty trace, $\langle\rangle$, is always a possible trace of the system. Since a trace without any Low events and purged of High events (observed as $\langle\rangle$ by Low) is always possible, Low cannot be sure whether there have been any High actions as High may have done nothing. It would seem to follow that non-inference could be regarded as a special case of non-interference, and also that non-interference implies non-inference.

In fact, O'Halloran makes this observation himself, although the reasoning is not clear. The converse, however, is not true. Full details of CSP can be found in [83].

In [118], McLean defines several information flow properties as functions of "trace sets", or sequences of High and Low inputs and outputs. Non-inference is described as "the property that is satisfied by a trace set $\sigma$ if and only if it is closed under *purge*" [118]. The purge function effectively removes all High level events from a trace, which coincides with the function, $purge_H$, given above. For details of McLean's framework, the reader is referred to [118].

Recall that from the previous proofs, opacity on the predicate, $\phi$, as defined in Section 3.2.1, implies the Purge Closure Property (PCP) and vice versa, which is in agreement with McLean's work [118]. Therefore, opacity with respect to $\phi$ implies non-inference and vice versa [139]. It follows therefore that the PCP is equivalent to non-inference.

As shown earlier, non-interference implies a variation of opacity, but the converse does not hold. It would be interesting to investigate the relationship between non-interference, non-inference and opacity. This is reserved for future work.

### 3.2.4 Non-leakage as Initial Opacity

Introduced by von Oheimb in [185], non-leakage expresses confidentiality over a set of initial information. Informally, an observer, $u$, should not be able to distinguish between any two states $s, s'$, if, during the course of a trace, $t$, caused by an action $a$ from $s, s'$, no domains have been allowed to interfere directly or indirectly with $u$.

By assigning actions to domains, the definition captures intransitive information flow. If this is not the case, or if individual actions are not distinguished, then weak non-leakage can be applied. Intuitively, for any action sequence of length $\alpha$, $u$ may only be influenced by specified domains up to a chain length $\alpha$. Weak transitive non-leakage may be appropriate if actions are not assigned to domains and $\alpha$ is not of interest. Essentially, if $u$ cannot distinguish between two initial states, then $u$ is unable to do so after any sequence of actions. Full details can be found in [185].

There is currently no formal definition of intransitive flow in the opacity framework, and it is assumed only that no information should flow from High to Low levels. This being the case, a special case of non-leakage, that is, defined for transitive flow and deterministic systems, can be cast as a variation of opacity. More precisely, with the predicate, $\phi$, defined as being true if it contains any initial states, we can model a system which satisfies opacity with respect to a set of initially secret information. The following formalises the relationship between a special case of non-leakage and this variation of opacity.

Suppose a system, $S$, which has a set of initial states, $S_0$. $S$ is modelled in terms of another system, $\hat{S}$, which has a unique initial state, and a set of distinct events, $I$, corresponding to the initial states of $S$. Formally,

$S \approx \hat{S} \setminus \{I\}$

This states that $S$ is indistinguishable from $\hat{S}$ with the initialising events hidden. In the spirit of CSP, $\hat{S}$ is a convenient device by which the initial states in $S_0$ can be identified using only $I$ events. Note that "$\approx$" is a suitable process equivalence, for example,

$$S \approx Q \Leftrightarrow S \setminus H \sim_{traces} Q \setminus H$$

**Definition 7** *$S$ is defined as non-leaking if*
$\forall\ i,j \in I \cdot i \sim j \Leftrightarrow \hat{S}/\langle i\rangle \approx \hat{S}/\langle j\rangle$,
*where* $S/\langle i\rangle := \hat{S}/\langle j\rangle$

Observe that the events $i,j$ have not been specified as either High or Low-level, so the equivalence holds for all observers, including High. This follows von Oheimb's definition [185], but may sometimes be too strong a requirement. For example, initially secret cryptographic key information may not need to be kept hidden from High-level users. Should High eventually learn this information and pass it to Low, it is too late for it to be useful.

For a deterministic system, $\hat{S}$, "$\approx$" in the definition can denote trace equivalence. Otherwise, as discussed previously, other forms of equivalence may be necessary to deal with non-determinism in the system.

Below, a special case of opacity is defined [139], in which the initial states are opaque, that is, $\forall i \in S,\ \phi_i$ is opaque.

**Definition 8** *Let $\phi_i(tr) \Leftrightarrow \nexists\, t$ such that $tr = i ^\frown t$, that is, $\phi_i$ is true of a trace $tr$ if $tr$ does not start with event $i$.*

*The initial states of $S$ are opaque with respect to $\phi_i$ and obs if and only if*
$\forall i,j \in S_0, \forall\, tr \in \tau(S/\langle i\rangle)$
$\exists\, tr' \in \tau(S/\langle j\rangle)$ *such that* $\phi_i(tr') = F \wedge obs(tr) = obs(tr')$,

*where the obs function is defined by*
$obs(e) = e$ *if* $e \in L$
$obs(e) = \langle\rangle$ *otherwise, i.e. if* $e \in H$.

The observation function here is the trivial in the sense that it is the identity on $\alpha S$, the alphabet of $S$. In other words, no events are hidden or identified. This would seem to be in accord with von Oheimb's definition, but it may be useful to extend this notion with generalisations in future work.

It should be noted that, in this work, I refer to specialisations of both non-leakage and opacity, that is, transitive non-leakage, and opacity over the set of initial events [139].

Transitive non-leakage can now be captured neatly as an opacity property. To do so, it is shown below that transitive non-leakage implies opacity on $\phi_i$, that is, Definition 7 $\Leftrightarrow$ Definition 8 [139].

*Proof.*

$\Rightarrow$

Suppose $S$ satisfies Definition 7. Definition 7 is equivalent to

$\forall i,j,\ i \sim j \Rightarrow \forall\, tr \in \tau(S/\langle i\rangle)$

$\exists\, tr' \in \tau(S/\langle j\rangle)$ such that $tr \upharpoonright L = tr' \upharpoonright L$

Note that $tr \upharpoonright L = tr' \upharpoonright L \Leftrightarrow obs(tr) = obs(tr')$

The equivalence of this definition with Definition 7 [139] is now quite clear. Both are asserting that for $i, j$, and for any trace of $S/\langle i\rangle$ there must be a trace of $S/\langle j\rangle$ that is indistinguishable to Low.

$\square$

Non-influence, also introduced in [185], is defined as non-leakage plus non-interference. von Oheimb observes that non-leakage could be regarded as a special case of non-interference where only the secrecy of initial events is important. Similarly, non-interference could be regarded as special case of non-influence where secrecy of initial events are not of concern [185]. However, he considers all three properties best kept separate, as non-interference and non-leakage are simpler than non-influence, and both are useful depending on application requirements. Likewise, the special case of opacity, defined above, is useful under certain circumstances.

Although it has been shown that non-leakage can be cast as a variation of opacity, the same is not true of non-interference. Therefore, there is no straightforward relationship between non-influence and opacity. Note that opacity over initial states is termed initial-opacity in [25].

In the next section I discuss some related work, and consider any impact they may have on future research.

## 3.3 Related Work

Mazaré [108] used opacity to describe the inability of an observer to distinguish between two encrypted messages. He has since worked with Bryans et al to create a formal framework for opacity [25], [26], [27], the essence of which has been discussed in this chapter. However, Mazaré's work remains distinct in that it mainly concerns the modelling and verification of cryptographic protocols [96], [108].

As described in Chapter 2, Hughes et al [84] define various forms of opaqueness on function views, which represent an observer's partial knowledge of a system. One motivation for their framework, was to provide an interface between a system specification and property specification, in particular utilising the relative strengths of process algebra and logics for the respective tasks. Although it has roots in epistemic logic [81], the function view belongs to domain theory [10], and the framework can apparently be applied to any process algebra and any logic

Function views and opaqueness can be expressed as predicates over observational equivalence, hence enabling the translation between logic and process algebra for verification [84]. This also indicates a link to the notion of opacity, which is also based on observational equivalence. However, also as noted earlier, aside from a conceptual similarity, the function view framework differs from that of opacity in its structure and logic base.

Function views appear to be particularly well suited to formalising properties based on relationships between entities, and this is demonstrated by a case study on anonymity. The framework is expressive enough to capture an impressively wide range of anonymity properties, including pseudonymity and privacy [84]. Defining different forms of anonymity in the opacity framework is a subject of future research.

The intuitive similarity between secrecy [78], [77] and opacity has also been noted. Recall that a predicate $\varphi$ is secret if agent $a$ never knows that $\varphi$ is true. However, unlike opacity, secrecy is based on epistemic logic and is not expressed in terms of observational equivalence.

Secrecy is based on a synchronous system, another distinguishing feature from the opacity framework which is asynchronous. Halpern et al define a run as a function from time to global states, and secrecy properties are defined using points, that is, in terms of the tuples $(r, m)$, where $r$ is a run at time $m$. In the case of opacity, a run could be regarded as an execution from a state to another possible state, and the property is defined without reference to time points.

An agent with "perfect recall" [78] can store all local state sequences at all times. A synchronous form of opacity could be obtained by encoding time in possible system states. Hence, there may be a link between "perfect recall" and the orwellian *obs* function [25]. Likewise, with "resource-boundedness" [77], [78] and the $m$-orwellian *obs* function [25]. Orwellian functions correspond to an observer with a potentially infinite memory to store labels, and can use either subsequent or previous knowledge of labels to (re-)interpret a label. $m$-Orwellian functions are a restricted class of orwellian, where the observer can only store a bounded number, $m$, of labels. Further details can be found in [25].

It is interesting to note that Halpern et al use synchrony in a slightly different sense to that, for example, in [118] and [186], where a synchronised trace is a (possibly infinite) sequence of events in a specified order, rather than a trace that is constrained by a time point. However, a "point" could be regarded as a finite trace, and Halpern et al show that translating either separability [118] or generalised non-interference [118] into a synchronous setting based on time points, weakens these two respective properties [77]. It is also worth noting that the original motivation for synchronised traces in [118] and [186] was for compositionality. This aspect has, as yet, not been dealt with for secrecy.

Probabilistic secrecy has been defined as an extension of the possibilistic form [78]. Lakhnech et al [96] have defined probabilistic opacity, but in a cryptographic setting. They claim that it is straightforward to apply their definition to general opacity, but this has yet to be investigated.

## 3.4 Discussion and Summary

Formal definitions of opacity [25] have been presented. Several existing security properties were then discussed in relation to opacity. It has been shown that non-interference implies a variation of opacity, but that the converse is not true [139] . As intuition might suggest, this indicates that non-interference is a stronger property than opacity.

Non-deducibility and (a special case of) non-leakage have also been cast as opacity [139]. However, non-influence, is defined as non-interference plus non-leakage [185]. As there is no direct relationship between non-interference and opacity, the same is true of non-influence and opacity.

It is important to note that non-leakage can capture intransitive flow policies if system events are assigned to domains, while opacity is currently limited to transitive information flow. As it has been shown that with non-leakage, the intransitive collapses to the transitive case [185], it was not problematic mapping it to the variation of opacity as described above.

The Purge Closure Property (PCP) [139], defined above, states that for all traces of the system, $S$, the purge of a trace is a valid trace of $S$, where "purge" means that all High level events have been removed. Recall that opacity implies the PCP and vice versa. The PCP is similar to the *purge* function defined by Mclean, and from his observation that a system satisfies non-inference if it is closed under *purge*, it can be concluded that opacity implies the non-inference and vice versa [139] .

Recall from Chapter 2, that a possible relationship may also exist between opacity and the Perfect Security Property [187].

The results of this chapter demonstrate the great flexibility of opacity. This is mainly due to the fact that it is a parametric property in that the predicate can be specified according to the requirements of the system. For example, as in much of the work above, opacity can be defined over the set of High-level events, or, as with non-leakage, the initial states of the system.

If a link exists between any two properties in question, it is possible that one of them can be expressed more succinctly if cast in terms of another property. Halpern et al, for example, state

$f$-secrecy as non-deducibility, showing that there appears to be a relationship between the two properties [77]. In subsequent chapters, I describe a case study of voting systems, in which opacity is used in the study of information flow. An interesting outcome was that opacity appears to be just the right tool to model properties such as coercion-resistance [91], [46], [38]. Furthermore, its parametric quality gives opacity the flexibility to adapt to the security requirements of different voting systems.

Information flow is extremely difficult to define, and the work discussed so far has highlighted the subtleties and complexities involved. This work has also tested the flexibility and versatility of opacity has been tested, and is a first step at establishing its distinct character. This is emphasised in the second part of my thesis, which demonstrates the complexities of information flow analysis. Together with some possible future extensions, to be discussed later, it is hoped that this work in its entirety will contribute towards a richer, more useful opacity framework.

# Chapter 4

# Voting Systems - A Case Study

This chapter begins an investigation of information flow in voting systems. The aim is to determine, firstly, the security requirements of a system, and secondly, where opacity can be usefully applied.

The work centres on Prêt à Voter [36], but several other voting schemes are also studied in order to make a comparison and to draw informed conclusions from the results obtained. In particular, the Chaum [35], Neff [131], [132], [133] and FOO [59] schemes are examined alongside Prêt à Voter. They all aim to achieve similar security properties, but differ quite widely in operation and the level of voter interaction required. Hence, as one might expect, their security requirements, and also their vulnerabilities, differ.

Interesting results were obtained from an analysis of the Chaum and Neff schemes by Karlof et al [92], in which they take the "systems perspective", and consider how possible interactions could subvert the protocol in each case. A similar analysis of Prêt à Voter [163] extends this work by identifying further vulnerabilities. This form of analysis will be discussed in Chapter 6. However, to set the scene, I first give a general overview of voting schemes, highlighting current trends and their aims. This is followed by descriptions of Prêt à Voter, and the Chaum, Neff and FOO schemes.

In the next chapter, I carry out an analysis of information flow in Prêt à Voter. The protocol is examined at design level, with less emphasis on interaction with the environment than the analyses performed in [92] and [138]. The aim is to determine the information flow properties required to ensure correct behaviour of the system, and, in particular, the extent to which opacity can be applied for this purpose. This is compared with a similar analysis of the FOO scheme, which differs form Prêt à Voter quite significantly in protocol execution and the type of interaction required of the voter.

Following this, is a chapter on the systems-based analysis of Prêt à Voter, which, as will be seen, has provided some interesting and elucidating results. I then investigate coercion-resistance, which is an important property, as it safeguards a number of requirements of voting system, such as ballot secrecy and eligibility. These requirements are discussed further shortly. Recent work on coercion-resistance [46], [91] or receipt-freeness [136] does not appear to present an easy way to formalise the respective properties, for example, in system specification. These definitions are somewhat cumbersome, and, in the case of the latter, make use of complicated mathematics and highly specialised cryptographic primitives. However, it will be shown that coercion-resistance can be expressed informally as opacity. A formal definition for receipt-freeness cast as opacity is proposed, and possible ways to reinforce the definition for coercion-resistance are discussed. These provide succinct and viable alternatives to other proposals. In addition, opacity may be linked to certain cryptographic primitives, such as Designated Verifier Proofs [88] and Deniable Encryption [30]. This is not formalised as it is beyond the present scope, but would be an interesting extension for future work.

The case study indicates that opacity is highly useful in the context of voting systems, and also that it may have possible application in cryptography. I reserve the overall conclusions for the final chapter.

## 4.1 An Overview of Voting Systems

Voting is the bedrock of democratic societies, dating back several millennia, and, as one might expect, a great many voting systems exist. These range from marked clay balls, used by the Ancient Greeks, to paper ballots that have persisted from ancient Roman times to the present day. Mechanisms have also been developed that attempt to ensure election integrity and handle large numbers of voters. The earliest is probably a lever-operated machine, first used in the United States in the 19th Century. A fairly comprehensive history of voting systems can be found in [90]. Cryptographic voting schemes, proposed in recent years, also offer highly desirable security properties, as well as minimal dependence on components. Some of them will be discussed in more detail shortly.

In addition to the many voting schemes in existence, there are also numerous voting methods [7], for example, using ranked pairs in Condorcet voting, single transferrable votes, designed to provide proportional representation, and the more straightforward "first past the post", familiar to current UK voters.

Although varying quite dramatically, voting systems tend to converge on a similar set of goals. These include [94], [59]:

- Ballot secrecy: only the voter should know how she voted.
- Legitimacy: only registered voters may vote.
- Eligibility: a voter may vote at most once.
- Individual verifiability: the voter should be able to check that her vote is accurately recorded for tabulation.

- Universal verifiability: the final tally should be verifiable by any third party.
- Accuracy: the final tally should reflect the true count of all legitimate, cast votes.
- Coercion resistance: a coercer should not be able to determine how a voter cast their vote, even with the co-operation of the voter. In other words, the voter should not be able to prove how she voted to a third party.

A voting system may, however, only satisfy these requirements to a certain extent. Due to vulnerabilities introduced by faulty software, hardware, interaction between system components or environmental factors, one or more requirements may not be satisfied at all. In certain contexts, some of the properties may not be necessary. For example, secrecy is often not required in parliamentary referenda.

In an attempt to improve efficiency and accessability of the election process, governments have invested in various automated voting systems that increase speed and accuracy of ballot counting. Arguably, some of these new mechanisms offer greater secrecy, and in the case of remote systems, increased voter participation. However, these attempts have been fraught with problems, many of which are due to reliance on computer hardware and software performing as intended [17], [6], [93], [100]).

Recent proposals for cryptographic voting systems promise to resolve some of these problems by introducing transparency into the voting process, that is, the protocols can be verified at intermediate points. Notable examples are Prêt à Voter [36], and the Chaum [35] and Neff [131], [132], [133] schemes. These systems aim to provide assurance of secrecy and accuracy from a high degree of transparency in the vote recording and counting stages, rather than over-dependence on the underlying technical system.

## 4.2 The Chaum and Neff Voting Schemes

Although cryptographically quite distinct, the Chaum and Neff schemes have a basic similarity in that the voter engages in a "cut-and-choose" protocol with a voting device. Once a vote has been cast, the device issues a receipt to the voter and transmits a copy for processing. It is convenient to describe them together, but pointing out the important differences. This is useful for the ensuing discussion of the systems-based analysis of the Chaum and Neff schemes, which was performed by Karlof et al [92]. Furthermore, due to similarities in protocol execution, the two schemes were found to have certain vulnerabilities in common.

The cryptographic detail of both the Chaum and Neff schemes, as with Prêt à Voter and the FOO scheme, are complex and unnecessary in the present context. Hence, I summarise the only key features of each scheme and refer the reader to the appropriate texts for further detail.

For both the Chaum and Neff schemes, an outline of the protocol is as follows. It is assumed that an authority has undertaken the pre-election set-up, for example, generating and certifying public keys as appropriate. Once authenticated at a polling station, the voter engages in the first stage of casting a vote, which is receipt preparation. This takes place in a booth where the voter interacts with a voting machine. The device issues an encrypted receipt, and the voter is given the opportunity to verify that it correctly encodes her vote. The receipt is posted to a ***Web bulletin board*** together with an identifier generated by the voting device, which is called the ***Ballot Sequence Number*** (BSN) in [92]. The voter retains a copy of their encrypted ballot receipt, which she can subsequently check on the Web bulletin board to make sure that the receipt has been posted correctly.

Once the voting and receipt verification phases have ended, the BSNs are stripped off, and the receipts shuffled and decrypted in a series of anonymising mixes to ensure that no link remains between the encrypted ballot receipts and final decrypted vote values. The results of each stage of the mix process are posted to the Web bulletin board. Intermediate stages of the mix can be subjected to, for example, random, partial auditing [87], which detects any attempted fraud during the mix and decryption process to a high probability. This is described later in the chapter.

Aside from the details of cryptographic primitives involved, and the way that the anonymising mixes are performed and verified, the essential difference between the two schemes is in the format of the receipt. As mentioned above, both schemes employ "cut-and-choose" protocols which allow the voter to detect any attempts by the vote recording device to encode their selection incorrectly in the receipt.

In the Neff scheme, after the voter selects a candidate, the device constructs a $n \times l$ matrix of *ballot mark pairs* (BMPs), which are El Gamal [60] ciphertexts of the digits "0" or '1", where $n$ is the number of candidates and $l$ a security parameter. Each row of the matrix corresponds to a candidate, and in the case of a chosen candidate, the BMPs encode either "00" or "11", in the unchosen candidate rows, either '01' or '10'. This matrix is called the *verifiable choice* (VC). The voter then chooses either a "basic" or "detailed" receipt. A basic receipt is of the form (BSN, hash(VC)), which the voter can use later to check that her receipt has been correctly transmitted. However, given the possibility of cheating devices, the voter can also verify the construction of the VC, and obtain a detailed receipt. To do this, the voting machine issues a pledge bit, $p$, and the voter, a challenge, $c_i$, for the chosen candidate row, $i$. The device "opens" each BMP in row $i$ of the matrix, by decrypting either the left or right BMP according to $c_i$, and the voter checks that it matches $p$. This is repeated for each row, the device opening the BMPs according to a random challenge, creating an *opened verifiable choice* (OVC) in the process. If the chosen row, $i$, has been incorrectly constructed, it will be detected with the probability of $(\frac{1}{2})^l$. To prevent coercion, the voter has the option of issuing challenges for the unchosen rows as well, overriding the machine's random selection.

In the Chaum scheme, the voter's choice is represented using visual cryptography [130] to generate a ballot image. The image is split between two encrypted layers, each of which, viewed separately, comprises random pixel patterns. The image can only be reconstructed by correctly superimposing the two layers. The voter selects either the top or bottom layer to retain as a receipt, so without knowledge of the encryption, the receipt does not reveal the voter's choice.

An example is shown in Figure 4.1 below [28].

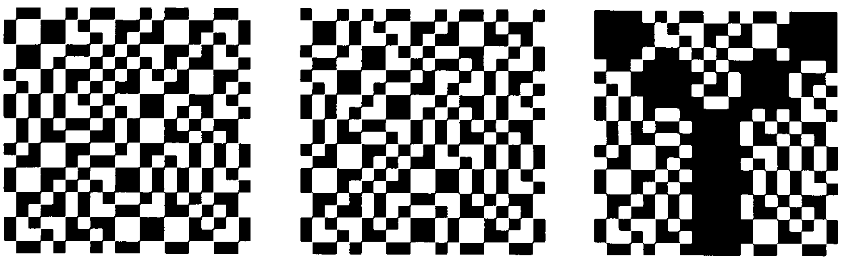

Figure 4.1: The Chaum Scheme - Formation of a Ballot Image

The voting machine also prints other information on the transparent layers, in particular the two "dolls", $D_l, D_b$, which carry details on how to decrypt, respectively, the top or bottom layers. The details are encoded by successive encryptions, each time with the public keys corresponding to a single stage of the mix. Outside the booth, the voter can perform well-formedness checks on her receipt, in particular, that the two dolls match. Any attempt by the device to cheat by incorrectly encoding the voter's selection in the receipt can then be detected. The importance of this step will be explained in more detail later.

A description of the mix process in the Chaum scheme, which is similar to the one currently used in Prêt à Voter, now follows. During the mix, pairs of encrypted vote and the corresponding dolls are posted to the tellers in batches. At each stage of the mix and for each pair, the teller strips off the outermost layer of the "doll". The teller performs two sets of shuffling on the pairs before posting to the next teller, who repeats the process. Final decryption of the "dolls" by the last teller reveals the information needed to recover the vote values. Figure 4.2 shows a batch posted to $n^{th}$ teller, the two sets of permutations, and the shuffled batch being posted to the $n - 1^{th}$ teller. Recall that the BSNs are removed before posting to the mix. Shuffling of the pairs destroys the order in which the receipts were posted on the Web bulletin board, and so removes any link between the BSN and corresponding encrypted vote.

To verify the decryptions performed by the tellers, the Chaum scheme utilises *randomised partial checking* [87] to audit the mix process. At each stage, the tellers are requested to reveal a randomly selected half of the links to the first shuffle, and another randomly selected half for the second shuffle. The two sets of selected links are disjoint so that a complete path cannot be traced between them.

This is illustrated in Figure 4.2 [28].

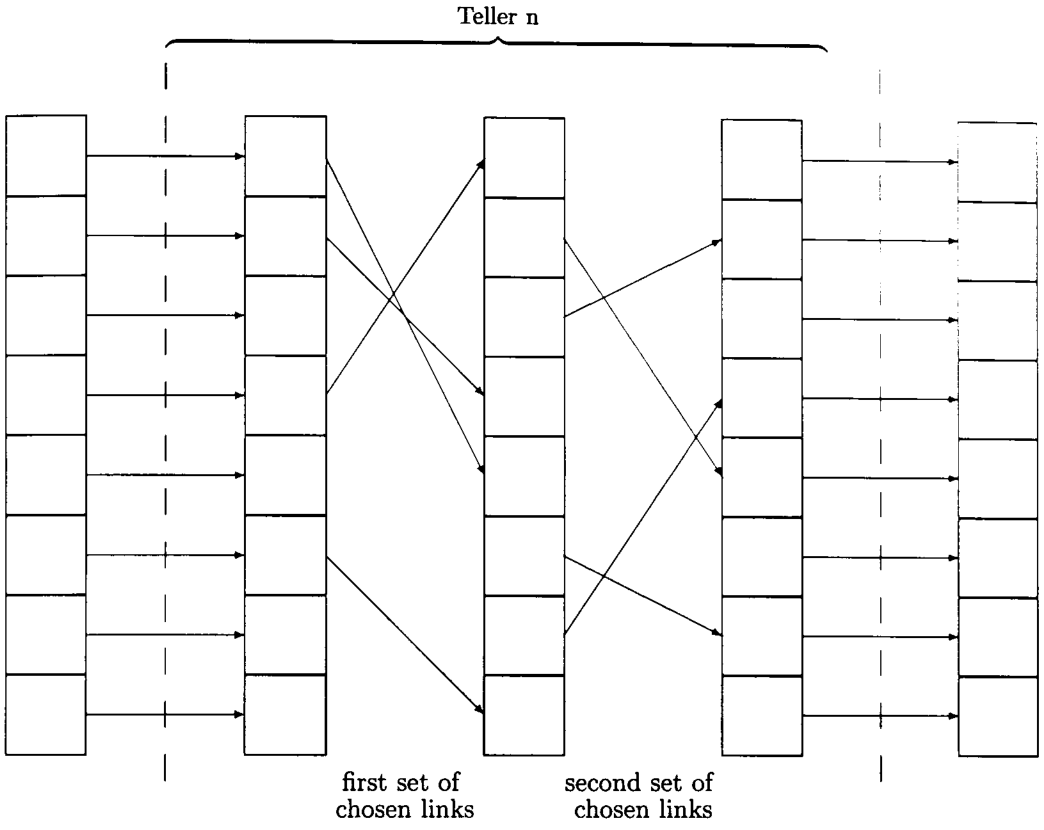

Figure 4.2: Revealed links from teller $n$'s permutations

The auditors can use the revealed links to check that the decryptions are correct. The mathematical details for this are not needed for the present work, but can be found in [35] or [28]. It should be noted that robustness of the mix process is implemented differently in the Neff scheme. The details are omitted, but the interested reader is referred to [133], [132], [131].

In both the Chaum and Neff schemes (and also Prêt à Voter), although anyone can verify that the final tally accurately reflects the votes cast, the voter cannot directly check that her vote value has been correctly decrypted. This is essential in order to avoid the possibility of coercion or vote buying. Auditing of the mix process, should however, provide some assurance that the decryptions are correct.

## 4.3 Prêt à Voter

I now give an overview of Prêt à Voter [36], which is a slightly modified and updated version of the one presented in [163]. Prêt à Voter is derived from the Chaum scheme, but uses a completely different mechanism to represent the encrypted vote value in the ballot receipt. In place of the visual cryptographic techniques of the Chaum original, the voters are provided with a ballot form. similar to the ones currently used in many paper-based systems. The following diagram illustrates an example.

| | |
|---|---|
| Nihilist | |
| Buddhist | |
| Anarchist | |
| Sophist | |
| | $7rJ94K$ |

Figure 4.3: A Prêt à Voter ballot form

The voter makes her selection in the conventional way, by placing a cross against the candidate of choice. A vote for the Sophist candidate is shown in Figure 4.4.

| | |
|---|---|
| Nihilist | |
| Buddhist | |
| Anarchist | |
| Sophist | X |
| | $7rJ94K$ |

Figure 4.4: A Ballot Form with a Marked Vote

To cast the vote, the voter separates the right- and left-hand strips. The left-hand strip should be destroyed, for example, by feeding it into a shredder. The right-hand strip is placed under an optical reader, which records the information on the right-hand strip, that is, the value at the bottom of the strip and the position of the "$X$". To be more precise, it is the numerical representation of the cell containing the "$X$" which is recorded. The role of the value "$7rJ94K$" is explained shortly. The right-hand strip is then returned to the voter to retain as her receipt, an example of which follows.

| |
|---|
| |
| |
| |
| X |
| $7rJ94K$ |

Figure 4.5: A Prêt à Voter receipt

To ensure authenticity, the ballot forms would be augmented with various anti-counterfeiting devices, and a digital signature applied to the receipt when the vote is cast.

Aside from the retention of a receipt, the process of casting a vote should be familiar, at least to a UK voter. One might think that possession of a receipt would open up the possibility of coercion or vote-buying. However, the candidate lists are randomised against a base ordering on the ballot forms. For example, with the following canonical ordering,

the ballot form shown in Figure 4.3 has an offset of 2, which is encoded in the value "7rJ94K" at the bottom right-hand side of the form. Known as the "onion", this value represents the offset

| |
|---|
| Anarchist |
| Sophist |
| Nihilist |
| Buddhist |

Figure 4.6: Base Order of Candidate List

which has been successively encrypted under the secret keys of a number of tellers. This is similar to construction of the "dolls" in the Chaum scheme, as described previously.

Once decrypted, the numerical value of the marked vote, in this case, 4, with the offset removed, yields the value 2, which translates as a vote for Sophist. Clearly, as long as the left-hand strip is removed, the right-hand strip alone does not indicate which way the vote was cast. One way of thinking about this is that, in contrast to other voter-verifiable schemes, the vote value is not directly encrypted to form the receipt. Instead, the (randomised) frame of reference, that is, the order of the candidate list against which the vote is represented, is encrypted. Another feature is that only the tellers acting in consort are able to reconstruct the candidate order and recover the vote value encoded in the receipt.

Once all voting has ceased, the receipts are transmitted to a central tabulation server which posts them to a secure Web Bulletin Board. This is a write-only, publicly visible facility. Only the tabulation server can write to this and, once written, anything posted to it will remain unchanged. Voters can visit the Web bulletin board and confirm that their receipt appears correctly. After a suitable period allowing voters to do this, the set of tellers take over and perform a robust, anonymising, decryption mix on the batch of posted receipts. The mechanism for this is similar to that used in the Chaum scheme, as described previously.

If all the steps are performed faithfully and the entities engaging in the process are trustworthy, one can be reasonably confident that the election will be accurate and that the vote values are kept secret. However, the aim of schemes like Prêt à Voter, and the ones devised by Chaum and Neff, is to achieve these goals without the need to place such trust in any of the components of the system. The mechanisms used to detect any malfunction or misbehaviour by the devices or processes that comprise the Prêt à Voter scheme are outlined below. Ignoring, for the present, any problems that might arise in the surrounding system, there are essentially three places where things could go wrong with respect to the accuracy requirement.

- The ballot form might be incorrectly constructed and the information encoded in the onion might not correspond to the candidate order shown on the left-hand strip. Subsequent decryption of the receipt would then reveal a different vote value to the one selected by the voter.
- The receipt might be incorrectly recorded or transmitted to the Web bulletin board.

- The tellers might fail to correctly decrypt the receipts.

There are two ways in which construction of the ballot forms can be checked. The first is through a random audit performed by independent entities prior to the election. Using the tellers to strip of the layers of encryption, the cryptographic material can be recovered, and the offsets for the forms re-computed. In addition, voters can cast "dummy" votes. For each trial, the tellers return the value of the vote, and, if correct, the voter can be reasonably assured that the form is properly constructed. For both types of check, it is important that the used forms are discarded, as the cryptographic material or votes have been revealed. Test voting could be carried out in the presence of an official, who subsequently enforces the destruction of used ballot forms. I return this issue later.

Once a vote has been cast, the voter can use her receipt to check the Web bulletin board, and make sure that it has been correctly transmitted and recorded. It is important that the ballot forms are authenticated, as this prevents dishonest voters from disproving the system. Conversely, voters are protected against a dishonest system. The Web bulletin board should also be secure, for reasons to be explained in Chapter 6.

As previously mentioned, Prêt à Voter incorporates randomised partial checking [87] to ensure that the votes have a high probability of being correctly decrypted. Full details of Prêt à Voter, including construction of the ballot forms, can be found in [36].

## 4.4 The FOO Scheme

Fujoika et al [59] devised the voting scheme which has become known as FOO. Like the Chaum and Neff schemes, it requires voter participation in a cryptographic protocol, but it differs in that there is no "cut-and-choose" element. Unlike the Chaum and Neff schemes and Prêt à Voter, the voter does not receive a physical receipt. In addition, the voter can be remote from the system, whereas the other three schemes currently require voter interaction at a polling station.

The key players in the scheme are the voters, an administrator and a collector, and there are three stages. In the first stage, the voter prepares a ballot and sends it to the administrator for verification and proof of eligibility. As explained shortly, blind signatures [33] are used to ensure ballot secrecy, and bit commitment [129] and digital signatures to ensure legitimacy and "unreusability" [59], that is, only legitimate voters may vote, and once only. It is assumed that all three cryptographic schemes cannot be broken [59]. The main steps are outlined as follows. Further details can be found in [59] or [94].

- The voter, $V$ computes a ballot $c = \xi(v, k)$ for a vote, $v$, using a bit commitment scheme, $\xi$, and a random key, $k_i$.

- V computes the message $m = \chi(c, f)$, using a blinding function, $\chi$, and a random blinding factor, $f$.
- $V$ sends her digitally signed message $\sigma_V(m)$ to the administrator, $A$. together with her identity.
- $A$ checks that $V$ has the right to vote, has not voted yet, and that her signature is valid. If all these are true, $A$ digitally signs $m$, returning $\sigma_A(m)$ to $V$.

In the next stage, the voter checks the ballot and casts her vote.

- $V$ unblinds the signed message, $\sigma_A(m)$, obtaining $b = \sigma_A(c)$, that is, the signed commitment to her vote.
- $V$ then sends $b$ on an anonymous channel to the collector, $C$
- $C$ checks that $A$'s signature is valid, and if so, enters $(l, b, c)$ on a list as the $l^{th}$ item.

After a certain period of time has elapsed, the collector starts the final stage.

- $C$ publishes the list of signed commitments, $(l_i, b_i, c_i)$.
- $V$ verifies her commitment, and sends $l, k$ to $C$ via an anonymous channel.
- $C$ then obtains $V$'s vote from the $l^{th}$ ballot, and publishes $v$.

The following diagram may help to clarify the scheme in terms of steps in the protocol:

$$V : c = \xi(v, k)$$

$$V : m = \chi(c, f)$$

$$V \rightarrow A : \sigma_V(m)$$

$$A \rightarrow V : \sigma_A(m)$$

$$V : b = \sigma_A(c)$$

$$V \rightarrow C : b$$

$$C \rightarrow BB : (l_i, b_i, c_i)$$

$$V \rightarrow C : (l_i, k)$$

$$C \rightarrow BB : v$$

where $V \rightarrow A : m$ denotes $V$ sends $m$ to $A$, $V : m$ denotes $V$ computes $m$, $BB$ is the bulletin board, and the remaining notation is as given above.

Note that the relationship between the ballot, $c$, and voter's key, $k$, are hidden by the blinding factor, $f$. As $c$ and $k$ are sent to the counter via an anonymous channel, the identity of the voter cannot be linked to the vote, $v$. In order to vote twice, the voter would have to obtain another valid signed vote, which can only be done by breaking the bit commitment scheme. Similarly, an illegitimate voter would have to break the digital signature scheme to obtain a valid ballot.

Kremer et al [94] have carried out a security analysis of FOO using the applied pi-calculus [8]. They report that FOO satisfies many of the requirements of voting schemes such as fairness, eligibility, secrecy and individual verifiability, that is, voters can verify that their votes were correctly recorded. However, they observe that, although not stated explicitly in [59], the protocol must be synchronised at two points to ensure fairness and secrecy. Firstly, registration must be completed before vote casting can take place, and secondly, the Collector must publish all the decrypted votes at once. If this does not happen, an attacker could, for example, block all registration until $V$ has cast her vote, and then link $V$ to her decrypted vote. Furthermore, early publishing of votes undermines fairness, as it amounts to release of partial results.

Another problem with the FOO scheme is that it is vulnerable to coercion. Although it is claimed to satisfy ballot secrecy, legitimacy an eligibility [59], [94], these requirements could also be at risk. I discuss coercion-resistance further in the next three chapters.

## 4.5 Discussion and Summary

In this chapter, I have discussed some of the properties required of voting systems in general, and given overviews of the Chaum, Neff and FOO schemes, and Prêt à Voter. Some of the potential weaknesses in each of the schemes are fairly apparent. For example, in both the Chaum scheme and Prêt à Voter, incorrectly decrypted votes and dishonest tellers during the mix process could be problematic. Some of these vulnerabilities are checked by design features, such as, in this example, randomised partial checking [87]. A systematic analysis may be necessary to identify other weaknesses. In the next chapter, I carry out an analysis of information flow in Prêt à Voter and the FOO scheme, with the aim of investigating whether variations of opacity can be used to capture their particular security requirements. As will be seen, the analysis also proved to be a useful way of locating information leaks in voting systems.

# Chapter 5

# Information Flow in Voting Systems

This, and the next two chapters are devoted to studying security properties in voting systems, in particular, Prêt à Voter. I begin by examining the interaction between participating entities in terms of information flow, and, as a comparison, repeat the process for the FOO scheme. The FOO scheme was chosen as it differs quite markedly from Prêt à Voter in several ways. Unlike Prêt à Voter, the voter in the FOO scheme engages in a cryptographic protocol with the system and does not receive a physical receipt. In addition, as will be examined later, FOO is vulnerable to coercion, whereas Prêt à Voter is fairly resistant to most forms of this attack. Another difference is that the FOO scheme can be operated remotely, whereas in Prêt à Voter [36], the voter casts her vote at a polling station. Informally, coercion-resistance can be defined as the inability of the voter to prove her vote to a coercer who may be able to influence certain choices made during the voting protocol. Note that coercion-resistance is almost impossible to guarantee in remote systems, as the attacker could be physically present when a vote is cast. This is discussed further in Chapter 7.

To analyse information flow in Prêt à Voter and the FOO scheme, I formulate a list of security requirements by examining various stages of each protocol. This also reveals the undesirable information flow that could occur at these stages, which is no less important, as awareness of both the desirable and undesirable information flow is useful in designing security assurance for a system. The various requirements are then translated into the appropriate existing security properties. As will be seen, many of the requirements can be expressed concisely in terms of opacity.

I first summarise the typical requirements of voting systems and discuss the extent to which they are satisfied, in each case, by Prêt à Voter and FOO. I then list a series security requirements for each scheme, taking into account both the desired and undesired information flows. This is followed by a translation of the requirements, where possible, into suitable security properties. Finally, I discuss the results.

## 5.1 Voting Systems - Requirements

In the previous chapter, several common requirements of voting systems were discussed. They include eligibility, accuracy, fairness, ballot secrecy, coercion resistance, and both individual and universal verifiability [94], [59]. Eligibility, or the ability to vote at most once, is often ensured by an authentication mechanism. In Prêt à Voter, as with the Chaum and Neff schemes, authentication is assumed. However, in [163], suggestions are made for possible defences against double voting in Prêt à Voter. These will be described in Chapter 6. In contrast, the FOO scheme includes a distinct authentication phase.

It is worth noting that absolute secrecy is a rather strong requirement, and is not always possible to achieve, for example, in the case of unanimous voting. As another example, consider the current UK voting system. After presenting a form of identification at the polling station, a voter receives a ballot form marked with a serial number that can be traced back to her entry in the register. The claim that tracing would only occur in exceptional circumstances, along with the sheer manual effort of finding a match for any one form amongst possibly thousands, should presumably reassure voters of their privacy. While a link to voter identity may sometimes be necessary, for example, in the case of disputed votes, the current trend is for the use of encryption to implement ballot secrecy, at the same time enabling verifiability [35], [131], [132], [36].

Ballot secrecy cannot be assured if the voting system is vulnerable to coercion. This can be a problem in schemes which make use of a publicly accessible bulletin board to allow voters to verify their receipts, as in, for example, [35], [131], [132], [36]. Note that, as previously mentioned, ballot secrecy is not always a requirement, for example, in shareholder voting and referenda. Once again, ballot secrecy fails if everyone votes the in same way.

Prêt à Voter is fairly robust against most forms of coercion, although it is potentially vulnerable to chain voting, which could be regarded as a form of coercion [163]. However, with suitable defences in place [163], and barring exceptional cases, such as everyone voting the same way, there can be reasonable assurance of ballot secrecy.

On the other hand, the FOO scheme is not coercion-resistant [59], [94]. This is discussed in more detail shortly. The scheme is claimed to maintain ballot secrecy [59], but vulnerability to coercion means that it could fail this requirement.

The FOO protocol requires three points of synchronisation, two of which partly concern secrecy, namely the administration and voting stages. If, for example, a voter, $V_1$, is able to vote before voter, $V_2$, has registered, then an attacker could deduce the identity of $V_1$ from the bulletin board, and hence, privacy fails. Furthermore, a corrupt administrator could block certain voters by not authenticating their votes, which could be regarded as a form of denial of service. The voting process must also be synchronised for the same reasons. A third point of synchronsation is necessary in the counting phase to ensure fairness, that is, to prevent early publishing of votes.

An administrative, or perhaps more precisely, authentication phase is not currently specified in Prêt à Voter. Although synchronisation in voting is not explicit, it should be possible to arrange for ballot receipts to be posted to the Web bulletin board only after the poll booths have closed. Alternatively, as suggested in [163], receipts could be posted in lexicographical order, for example, based on the onion value, thus avoiding possible attacks on voter privacy. To ensure fairness, publishing of decrypted votes could be delayed until all the receipts have gone through the mix process.

Another common requirement for voting systems is the possibility for anyone to check that the system is behaving correctly. In Prêt à Voter, this is achieved by auditing, for example, to ascertain the correctness of the mix process and well-formedness of the ballot forms. Both have been described in Chapter 4, but further details can be found in [36]. It has already been observed that neither voters nor independent third parties are able to verify the current UK voting system.

Both Prêt à Voter and the FOO scheme provide voter verifiability, that is, voters are able to check that their votes have been posted correctly. In the FOO scheme, voters actively verify the encryption of their votes. However, this feature, together with a publicly accessible bulletin board, lays the scheme open to coercion. All the voter has to do is supply the coercer with the decryption keys. To avoid this problem, most cryptographic voting schemes do not allow the voter to check that her vote was included in the final tally, that is, to identify her decrypted vote. However, as previously described in Prêt à Voter, auditing of the mix process should assure voters that the final tally is correct. The voter can also participate in the auditing of the ballot forms, which should provide some confidence that her vote is correctly encrypted.

In subsequent sections, I investigate how some of the properties discussed above may be specified in Prêt à Voter and the FOO scheme.

## 5.2 Prêt à Voter - Security Requirements

Prêt à Voter was described in Chapter 4 to provide the necessary background for the ensuing analyses. To begin, I formulate a series of requirements to ensure that the protocol operates securely and as intended. They are listed below.

1 A voter should only be able to vote once.

It has already been noted that Prêt à Voter does not currently implement authentication, although this could be incorporated into the scheme as described in [36]. The issue of "one voter, one vote" [94] is important for election integrity, and is included as a requirement.

2 Only the voter should know the value of her vote.

As discussed earlier, encrypted ballot receipts enable voter verifiability without, at the same time, enabling proof of a vote to a third party. However, as will be seen in the next chapter, encrypted receipts alone may not be an adequate defence against coercion, and additional counter-measures may be necessary.

3 The voting device should not be able to learn a voter's choice.

In Chaum and Neff schemes, there is a cryptographic exchange between the voter and the device in the creation of a ballot form. As shown in [92], there are opportunities for coercion if the device "learns" the vote value. One of the strengths of Prêt à Voter is that the voting device only scans the ballot receipts and does not "learn" the voter's choice [163]. With suitable implementation, the possibility of misuse of information by a malicious device is therefore reduced. This issue is discussed further in Chapter 6.

4 The web bulletin board should only display (encrypted, or anonymised and decrypted) legitimate votes.

The requirement that only legitimate receipts should be posted to the web bulletin board raises the issues of error detection, handling and recovery. These are subjects of ongoing research.

5 The voter should be able to check that her ballot receipt has been correctly posted.

The Web bulletin board should be securely implemented so that, for instance, it cannot be arranged for the voter to see her correct vote, when in fact it has been replaced with another value. This will be discussed in the next chapter.

6 Anyone should be able to verify that the final count is correct.

This requirement is ensured partially by public access to a Web bulletin board and auditing of the mix process.

7 The tellers' secret keys should not be revealed.

8 The permutations performed by the tellers during the mix process should not be revealed.

This is clearly to prevent encrypted votes being linked to the final decrypted values as they pass through the mix.

9 Votes should be correctly decrypted.

Correct decryptions are ensured, to a high probability, by randomised partial checking.

10 The algorithms used to check the well-formedness of the ballot forms should be public.

This is to allow auditing of the ballot forms by independent third parties.

11 For receipts assigned to either the right or left-hand partitions, tellers should reveal. respectively, the outgoing or incoming links and the corresponding cryptographic values as requested by the auditors. Once this has been done, the complement links should not be revealed.

This is done so that the auditors can verify the decryptions performed during the mix process.

The idea behind the mixing is to ensure that no links can be made between the encrypted ballot receipts and decrypted votes. In addition, randomised partial checking is designed so that a complete path cannot be traced through a mix. This has been described in the previous chapter. Full details can be found in [87].

12 It should not be possible to link receipts to either the partially or fully decrypted votes.

This is implemented by the mix process, and ensures ballot secrecy.

13 The left-hand strips should be destroyed.

Recall that if the voter manages to retain the corresponding half of her ballot receipt, she can then prove her vote to a third party. As discussed previously in Chapter 4, once a vote has been marked on the ballot form, the left-hand strip should be destroyed. Alternative measures to prevent coercion, such as making "dummy" left-hand strips freely available in the booths, have also been suggested [163]. I return to this issue in Chapter 6.

While every effort has been made to ensure the completeness of the above list, it is possible that other security requirements will be necessary with future enhancements of Prêt a Votèr. Conversely, some of the requirements above may no longer be necessary. The same comments apply to the list of security requirements for the FOO scheme which follows shortly.

## 5.3 Prêt a Votèr - Information Flow Properties

In the following, the requirements above are related to suitable information flow properties where possible.

The first Requirement involves authentication, which could be regarded as an enforcement, rather than information flow, property. More precisely, authentication in this case, ensures that a voter does not vote more than once. It could thus be regarded as a safety property as it prevents a "bad thing" from happening [172]. The difference between enforcement and information flow properties has been discussed previously in Chapter 2. Further details can be found in [172] and [161].

Likewise, Requirement 4 relates more closely to safety, rather than information flow properties. In other words, it requires error detection to identify erroneous votes, followed by error handling and recovery, which could be regarded as forms of enforcement.

Requirement 2 could be captured by an opacity statement, that is, with $\phi$ defined as being true of a trace if it contains the true value of a vote, and should be opaque to all except the voter. However, as the anonymised, decrypted vote is later displayed on the web bulletin board, this should perhaps be reinforced by a statement that voters should be anonymous. Interestingly, two different viewpoints are being considered. Firstly, that of the voter, and secondly, a potential coercer.

Requirement 7 and 8 could also be expressed using variations of opacity. In the former, $\phi$ could be defined being true of a trace if it contains the value of a teller's secret keys and should be opaque to all except the corresponding teller. In the latter, $\phi$ could be defined as being true if a trace if contains the permutations performed by a teller, and, again, should be opaque to all but the teller concerned. Clearly, this is to ensure that no receipts can be linked to the decrypted votes.

Requirement 10, however, is more complex. In order to verify the construction of the ballot forms, cryptographic material must be revealed. Once this has been done, the forms must be discarded, as ballot secrecy and coercion-resistance would be at risk if forms were re-used. Security of the system, however, is indirectly ensured by Requirement 2, that is, only the voter should know how she voted. Requirement 10 also ensures that the voter should be able to check the construction of ballot slips before casting a vote.

To express Requirement 12 in terms of opacity, $\phi$ could be defined as being true of a trace if it contains the links between the encrypted receipts and either the partially or fully decrypted votes and should be opaque to all. This relates to randomised partial checking, and is connected to Requirements 8 and 9 above.

Finally, Requirement 13 could be regarded as another safety property, that is, once the voter has marked her choice, the left hand strips should be destroyed.

Interestingly, Requirement 10 requires making certain information visible or accessible to all, rather than monitoring the flow of information. This could be regarded a form of liveness [12].

It would seem that many of the security properties in Prêt à Voter could be expressed in terms of a suitably defined form of the opacity property. However, it is interesting that a significant proportion of the security requirements apparently relate to safety and liveness properties. A possible reason for this is that ballot forms are prepared in advance, and perhaps less information flow control is required during protocol execution than there might be, for example, in a scheme which the voter is involved in ballot form creation. However, authority knowledge of ballot information has been identified as a vulnerability in the current Prêt à Voter, and control of information flow may be necessary during preparation of the ballot forms. A scheme for distributed ballot creation to address this problem is described in [165]. The main security concerns during a run of the current protocol involve correct recording of a legitimate vote from its representation as an encrypted receipt to its final publication in decrypted, anonymised form. Hence, many of the steps in the protocol relate to safety and liveness properties.

## 5.4 FOO - Security Requirements

There now follows a list of security requirements for the FOO voting scheme. Recall that the protocol, as described in the previous chapter, involves the voter, an administrator and a counter.

1 The key used to commit a vote should be known only to the voter. However, once the published ballot is verified, the voter should send the key to the counter to reveal the value of the vote.

2 The random factor used for blinding a committed vote should be known only to the voter.

3 The voter's secret key should be known only to the voter.

4 The voter's public key should be available to the administrator.

5 The administrator's secret key should be known only to the administrator.

6 The administrator's public key should be available to all voters and the counter.

7 Communication between the voter and counter should be anonymous.

8 The counter should authenticate all received ballots, assign a number to each one and publish the list on the bulletin board.

9 The bulletin board should be accessible to all.

10 The voter should check that her ballot is correctly entered on the bulletin board.

11 All decrypted votes should be included in the final tally.

Requirements 1 - 6 above involve security of the cryptographic keys used during the protocol, and are largely self-explanatory. As discussed in Chapter 4, the FOO scheme is based on the assumptions that the blind signature, bit commitment, digital signature schemes are unbreakable [59].

Another assumption is that an anonymous channel exists between the voter and counter. There is, however, no specification of the level of anonymity required. Dolev et al, for example, make a distinction between "untappable" and anonymous channels [49]. The former is a stronger requirement based on information theory [177], that is, the adversary should not be able to detect the sending or receiving of messages over the channel. In communication over the Internet, it may be difficult to trace point-to-point connections. Hence, an anonymous channel could, in practice, be regarded as untappable [91].

In [59], the authors note that voter checking of posted receipts is essential for verifiability. This means that integrity of the election result relies to an extent on voter diligence. In Prêt a Votèr, both auditors and voters can check the well-formedness of ballot forms before any voting takes place. In addition, with a VVPAT(voter-verifiable paper audit trail) [119]-style mechanism in place, an independent third party could reinforce voter checking of the posted receipts. This is discussed in further detail in the next chapter.

There is no mention of a mechanism in the FOO scheme to ensure that all votes are included in the final tally, or a strategy in the event of malformed or erroneous votes. It appears to be assumed that protocol runs as intended, and that the Counter is trusted to perform the publishing and final tally correctly. Error handling and recovery strategies are both important for the integrity and timeliness of the election, and although not currently implemented in Prêt a Votèr, are part of future research.

## 5.5 FOO - Information Flow Properties

It appears that variations of the opacity property can be used to express several of the requirements in FOO. Requirement 1 could be captured by defining $\phi$ as being true of a trace if it contains the value of the key used to commit a vote, and specifying that it should be opaque to all except the corresponding voter in the administration and voting phases. However, in the counting phase, the counter should also have access to the voter's key. The latter relates to Requirement 11.

To express Requirement 2 in terms of opacity, $\phi$ could be defined as being true of a trace if it contains the value of the random factor used for blinding a committed vote, which should be opaque to all except the voter concerned. For Requirement 3, $\phi$ could be defined as being true of a trace if it contains the value of the voter's secret key, which, likewise, should be opaque to all except the particular voter. In Requirement 5, $\phi$ could be defined as being true of a trace if it contains the value of the administrator's secret key, which should be opaque to all except the administrator.

In contrast, it seems clear that Requirement 7 relates to anonymity, that is, the communication between the voter and counter should be anonymous. This could be captured, for example, by a statement that the voter should be anonymous. Although an anonymous channel is not explicit in the scheme [59], it may be desirable that the voter's true identity should also be anonymous to the administrator. This could be achieved, for example, if authentication was based on a pseudonym, rather than the voter's true identity.

Interestingly several of the requirements appear to rely on liveness rather than information flow properties. For example, Requirement 4 requires the voter's public key to be available to the administrator, and could be regarded as a form of liveness. Requirements 6 and 9 are also related to availability: the former demands that the administrator's public key should be available to the voters and the counter, and the latter, that the bulletin board should be accessible to all.

Requirements 8, 10 and 11 all require certain actions to occur. that is. the counter should carry out her duties, that the voter should check her receipts and that all votes should be included in the final tally. These are, once again, perhaps best expressed as forms of liveness. An informal definition of liveness was given in Chapter 1, but a detailed discussion is beyond the present scope. The reader is referred to [12], [172], [161] are for further information.

## 5.6 Discussion and Summary

Prêt a Votèr and the FOO scheme have been analysed in terms of secure information flow. For each scheme, a list of security requirements have been formulated to ensure that the protocol operates with the intended security features. The requirements have then, where possible, been stated in terms of appropriate information flow properties, and it appears that variations of the opacity property can used in many cases. In the FOO scheme, communication between the voter and both the administrator and the counter, takes place via an anonymous channel, which could be expressed as forms of anonymity.

Interestingly, for both schemes, several security features rely on safety and liveness rather than information flow properties. Several of the security requirements in the FOO scheme relate to liveness, that is, certain events should occur. An example is that anyone should have access to the bulletin board, which is also a requirement in Prêt a Votèr. In addition, Prêt a Votèr has several security requirements which relate to safety properties, that is, certain events should not occur. An example is that illegitimate votes should not be published, which could be regarded as a safety property.

Although the security of both schemes appears to rely partially on liveness, and in the case of Prêt a Votèr, safety properties, many of the information flow requirements can be expressed in terms of opacity. It would seem that opacity strikes the right balance between security properties such as non-interference, which may be too strong, and non-deducibility, which may be too weak. In addition, the parametric nature of opacity gives it the flexibility to capture a range of different requirements. It is also convenient to work with a single property for information flow analysis such as this, which provides a uniform way of expressing security requirements.

To summarise the above results, it appears that requirements such as accuracy, fairness and eligibility, can be expressed as safety and liveness properties, whereas confidentiality can be captured with appropriately defined opacity statements. There may be alternative ways of expressing these requirements, but I reserve this for future work.

In the following chapter, and still on the theme of voting systems, I discuss another form of analysis, recently carried out by Karlof et al [92] and Ryan et al [163]. In contrast to the analysis described above, the emphasis is on interaction between the various components of the protocol. and takes environmental factors into account. Potential threats are considered at a different angle, that is, vulnerabilities due interaction with hardware and software, and also human fallibility, rather than insecure information flow at design level. It is possible that the two types of analysis may be useful in tandem, one revealing a possible attack on the protocol, where the other may fail. Further discussion and conclusions are, however, best deferred to the next two chapters.

# Chapter 6

# A System-based Analysis of Prêt à Voter

It is generally recognised that the security of even the best-designed technical systems can be undermined by environmental factors that violate, often implicit, assumptions. In addition, implementation flaws, faulty procedures and human fallibility can exacerbate security weaknesses. This is no less true of cryptographic voting systems, which have stringent security requirements but also, typically, a large user base and infrequent usage.

While it is important to analyse the core protocol in order to assess its security, it is also essential to consider the interaction between the various components of the system, for example, computer hardware and software, voters and voting officials. This point was argued by Ryan [162]. Adding to the complexity of these tasks, collusion between parties can make possible attacks more difficult to detect and resolve.

A protocol analysis was carried out on Prêt à Voter and the FOO schemes in the previous chapter. Similar work, but using different methods, has also been done on the Chaum [28] and FOO [94] schemes. In contrast, Karlof et al present a systems-based analysis of the Chaum and Neff schemes in [92].

A systems-based analysis could also be regarded as an analysis of information flow, but at a higher level than that carried out in the previous chapter. For example, vulnerability to chain-voting in Prêt à Voter is identified by both forms of analysis. Previously, this was done by considering possible information flow in the system, and also the different capabilities of the adversary. Chain-voting attacks in the analysis to follow, are identified by considering how the adversary might obtain an unmarked ballot form to initiate the attack. As mentioned earlier, a rigorous approach may be to carry out both forms of analysis, or combination of the two, on the system in question. Note that chain-voting is regarded as a form of coercion.

In this chapter, the work of Karlof et al [92] is extended with a systems-based analysis of Prêt à Voter [163]. Prêt à Voter was found to be remarkably robust against the vulnerabilities described in [92]. In addition, some further vulnerabilities and threats not mentioned in [92], are identified. An example is chain voting attacks, which do not apply to the Chaum or Neff schemes, but are a potential threat to Prêt à Voter. Where appropriate, enhancements and counter-measures are suggested.

Although important in itself, this analysis has also enabled a more justified comparison with one carried out previously in Chapter 5. This is discussed in the concluding summary.

## 6.1 Cryptographic Voting Protocols: Chinks in the Armour

The vulnerabilities described in [92] fall into four main categories: those due to subliminal channels, "social engineering" style attacks against the cryptographic protocols, denial of service attacks and implementation flaws. In this section, each of these vulnerabilities are described, and how they may apply to the Chaum and Neff schemes. Overviews of the Chaum and Neff schemes were given in Chapter 4, and so will not be repeated here. Full details of the schemes can be found in [35] and [131], [132], [133], respectively.

### 6.1.1 Subliminal Channels

Subliminal channels can arise whenever there are alternative valid encodings of the "intended" information. Additional information can be encoded by suitable choices between these alternatives. Public access to the Web bulletin board makes this a particularly virulent threat for voter-verifiable schemes. Two types of subliminal channels were discussed in [92]: *random* and *semantic*.

As described in [92], the Neff scheme makes use of randomised cryptographic primitives, namely El Gamal, in the creation of the ballot receipt, giving rise to a possible random subliminal channel. By judicious selection of random values, the voter's choice could be encoded in the encrypted receipt. Any agent with knowledge of the coding strategy for this subliminal channel, possibly in collusion with a malicious voting device, could then gather this information by observing the posted receipts.

Random subliminal channels are not a problem in the Chaum scheme, as it uses deterministic algorithms, namely RSA. However, *semantic* subliminal channels, can occur if there are alternative valid representations of the ballot image. Certain information could then be conveyed by alterations in the ballot image. Note that, in contrast to a random subliminal channel, information from a semantic channel emerges after the receipts have been decrypted and passed through anonymising mixes. It would therefore make sense for a malicious device to encode information about a voter's identity, rather than the vote value. In personal communication, Chaum expressed the view that semantic channels are not really subliminal, as they can be detected, that is, after decryption of the votes. However, the attacker may still achieve his goals if anomalies in vote representation are not detected by the authority before publishing to the Web bulletin board.

Subliminal channels could also give rise to kleptographic attacks, which were identified as a possible threat to Prêt à Voter [65]. This will be discussed later in the chapter. Note that there are other possible channels through which information could leak from the voting booth, such as, hidden wires, mobile telephones and wireless-enabled devices. These are not identified in [92], perhaps because they are regarded as inevitable. As will be made clear later, such channels are not a problem in Prêt à Voter.

#### 6.1.1.1 Mitigation

Counter-measures for random subliminal channels are problematic, since the randomness may be essential for some security properties. Karlof et al suggest that the same properties could be achieved using zero-knowledge proofs [68], but point out that most ZKP schemes also require randomness [92].

A possible approach is to require the use of pre-determined randomness, for example, from pre-generated tapes, as suggested by Neff in [92]. In effect, the non-determinism is resolved before the voter's choices are communicated to the system. The difficulty with this approach is ensuring that the devices adhere to this pre-determined entropy. In personal communication, Neff describes a recent implementation of his scheme, in which ballots are created in advance by a multi-authority process similar to a mix-net. The function of the device is completely deterministic, so there is no possibility of a subliminal channel. In addition, the voter only makes one random choice for the entire ballot, rather than for each option, as previously described in Chapter 4. Details of these innovations can be obtained from [5].

The Chaum scheme can be implemented so as to be fully deterministic, that is, using sequential ballot sequence numbers. The cryptographic seed material is then generated deterministically from the sequence numbers using the voting machine's signature. The validity of these signatures is randomly checked, at least on the layer chosen by the voter for retention. The machine's adherence to this requirement is then, in part, randomly verifiable. In addition, all posting to the Web bulletin board could be done in numerical order, starting with the encrypted ballot receipts to final decrypted votes. Hence, the shuffles applied by the tellers during the mix stages are pseudo-random and deterministic. Note that the teller's adherence to this requirement is fully and universally verifiable.

As previously described, the Chaum scheme, like Prêt à Voter, uses randomised partial checking for auditing the decryptions performed during the mix process. Each teller involved in the mix-net reveals only a proportion of input-output links, which are then checked to verify the decryptions. The selection of links to be opened for audit is constrained to ensure that no complete links can be traced through the net. Tellers should not know in advance which links to reveal, otherwise there is opportunity for undetected vote-altering. The selection of links for audit can in fact also be made deterministic by applying suitable cryptographic functions to the data posted [87]. This is not so much to counter the possibility of a subliminal channel, as the auditors should not have any sensitive information to leak, but rather to counter any possible collusion between the tellers and the auditors. Such collusion might allow a teller to know in advance that it will not be asked to reveal certain links and so would be able to corrupt those links without being caught.

Karlof et al [92] suggest trusted hardware as another possible mitigation against subliminal channels, for example, to enforce the voting machine's conformance with pre-determined entropy. This, however, as they note, is unsatisfactory, as the hardware and the software being used need verification and monitoring to detect any tampering. Furthermore, the idea is contrary to the principle of transparency and minimal dependence which the Chaum, Neff and Prêt à Voter schemes strive to achieve.

To prevent semantic subliminal channels in the Chaum scheme, one could enforce a standard format for each candidate option. However, this also runs into the difficulty of monitoring and enforcing adherence. It is also difficult to square with the possibility of "write-ins" that the Chaum scheme enables, in other words, allows the voter to specify her choice from an unbounded set of candidates. There may still be possibilities for subliminal channels, for example, the assignment of ballot sequence numbers, which should be monitored with care.

In the case of Prêt à Voter, an intrinsic feature of the design is that all non-determinism is resolved before the voter's choice is revealed. Furthermore, as the vote capture device does not "learn" the voter's choice, these vulnerabilities are eliminated. An interesting observation is that this relates to the loose-bisimulation definitions of non-interference [164], in that the High-level user cannot predict or influence system non-determinism in such a way as to communicate with Low.

### 6.1.2 Voter Participation in Cryptographic Protocols

Both the Chaum and Neff schemes require quite complex, multi-step interactions between the voter and the machine. As both schemes involve "cut-and-choose" protocols, the sequence of steps in the protocol can be highly significant. For example, in the Chaum scheme, it is essential that the device commits to the "dolls" before it knows which layer the voter will choose to retain. The protocols are designed so as to detect any attempt to construct receipts that do not encode the voter's true choice. The choice the voter makes in a "cut-and-choose" step will be referred to as the "protocol choice", to distinguish this from the voter's candidate choice.

By re-ordering the steps in the protocol, or introducing extra ones, the machine could learn the voter's protocol choice before it has to commit to the receipt encoding [92]. The device could then corrupt the vote value with impunity. Voters may not notice or appreciate the significance of such changes in the protocol execution. Similarly, the machine could feign an error and re-boot after it learns the voter's protocol choice. Relying on the voter making the same choice in the second round of the protocol, the machine then constructs a receipt for a different candidate. If the voter changes her mind, the machine re-boots again until the voter gets it "right" [92].

Another possible vulnerability of the Chaum scheme not identified in [92], but mentioned in [162], is as follows. The machine attempts to corrupt the vote value by incorrectly constructing one of the layers. If the voter chooses the other layer to retain as the receipt, the corruption will go undetected. However, if the voter chooses the corrupted layer, the attempt at deception could be detected. The machine could try to fool the voter by printing "destroy" instead of "retain" on the layer that the voter chose as the receipt. If the voter fails to notice this, or simply ignores it, the corruption will again pass undetected. This is another way that a corrupt machine could undermine the "cut-and-choose" element of the protocol. Even if the voter does notice the switch, and is confident that they are right, it may be difficult to prove this to a voting official.

#### 6.1.2.1 Mitigation

Many of the problems discussed above arise because the voters are required to participate in a protocol that they may be unfamiliar with, especially as voting is an infrequent activity. The security of the scheme depends on the voters following the steps and performing the relevant checks. In practice, it may be unreasonable to expect voters to spot any discrepancies in the execution or to appreciate their significance. Many otherwise correct technical systems are let down by a failure to take proper account of human fallibility. The tendency of Enigma cipher clerks to reuse the same message indicators and to use predictable preambles, is an excellent example.

The obvious approach is to try to ensure that voters understand the procedures and the appreciate their motivation [92]. This is similar to the notion of instilling a "security culture" in an organisation. In practice, this may not be feasible in the context of a rarely-used voting system with a potentially vast set of users. This limits the effectiveness of voter education in preventing protocols attacks, especially if the protocols are lengthy and complicated.

A further possible mitigation is parallel testing during the election, that is, for auditors to cast test votes and observe the behaviour of a device. This could also be problematic, as the auditors would need to be acutely familiar with the voting protocol. In addition, steps would be needed to prevent confusion between the test and real votes in such a way as to avoid signalling to the device any distinction between real and test votes. In any case, the machines could evade the auditing by altering the protocol only intermittently.

Arguably, a better solution is to make the voter experience much simpler. This is characteristic of Prêt à Voter, in which voters do not engage in a cryptographic protocol, and hence the scheme is not vulnerable the ensuing problems. This issue is discussed in further detail later in the chapter.

### 6.1.3 Denial of Service

Karlof et al [92] describe several ways in which denial of service attacks could disrupt or invalidate an election. For example, the voting device could falsify receipts, either duplicating them or submitting votes of its own choice. The former can be detected by officials scanning the Web bulletin board, but identifying the forged receipts is more difficult.

Other possible scenarios are for the devices, perhaps in collusion with an outsider, or tellers colluding with each other, to mount a denial of service attack. This could be global or more selective, for example, only in constituencies where the pattern of votes goes against some chosen candidate. In both the Chaum and Neff schemes, the machine necessarily "learns" the voter's choice during receipt construction. Hence, a selective denial of service attack, instigated by the devices, is a feasible threat.

For all these attacks, adequate error-handling and recovery strategies need to be in place. In addition, some form of back-up is desirable, for example, a *voter-verifiable paper audit trail* (VVPAT) [119], in which the authority retains paper copies of the ballot receipts should re-counting be necessary.

These attacks and counter-measures will be discussed further in the next section, when the robustness of Prêt à Voter is examined in more detail.

## 6.2 Systems-based Analysis of Prêt à Voter

In this section, Prêt à Voter is assessed for its resistance to the attacks identified in [92]. Prêt à Voter was previously described in Chapter 4. Full details can be found in [36].

### 6.2.1 Subliminal channels

Neither random nor semantic subliminal channels are a problem in Prêt à Voter. This is mainly due to the way that votes are encoded in Prêt à Voter. Many cryptographic voting schemes require the voter to supply her vote choice to the device, which then produces a verifiable encryption. In the case of Prêt à Voter, the voter's choice is encoded in a randomised frame of reference. It is the information that allows this frame of reference to be recovered, and so allowing the vote value to be recovered, that is encrypted. This can be done ahead of time, without needing to know the vote value. In Prêt à Voter, the ballot forms are generated in advance and allocated randomly to voters. Thus, the cryptographic commitments are made before any linkage to voter identities or vote choices are revealed.

Regarding semantic subliminal channels, suitable implementation should ensure that, for a given ballot form and vote choice, the digital representation in a Prêt à Voter receipt is unique, that is, the onion value and the number indicating the chosen cell on the left-hand column. Hence, there should be no possibility of a semantic subliminal channel. Even if the device were to learn the voter's choice, there would not be a means to communicate this information.

Like the Chaum scheme, Prêt à Voter [36] uses deterministic cryptographic algorithms. Encryption during ballot creation and decryption during the mix process are deterministic. In addition, the tellers can be required to post batches of transformed ballot receipts in lexical order at each stage. It is not clear that the latter is, in fact, necessary as the tellers do not have access to any privacy sensitive information.

In Prêt à Voter, the device does not "learn" the voter's choice. The device only scans the ballot receipts and posts digital copies of them to the Web bulletin board. Strictly speaking, we need to ensure that the device does not get sight of the candidate list. There are various ways to achieve this, but they have to balanced against the issue of enforcing the destruction of the left-hand strip, that is, the candidate list, to be discussed later in the chapter. As a result, in contrast to the Chuam and Neff schemes, other leakage channels from the booth, such as, hidden wires and wireless-enabled devices, cannot be exploited in Prêt à Voter. Even if a randomised primitive were to be used in the construction of the ballot form, any resulting subliminal channels would still be of no use to the adversary. This observation will be significant later, when future enhancements of Prêt à Voter, that utilise El Gamal encryption, are evaluated.

### 6.2.2 Protocol Attacks

In Prêt à Voter, as mentioned previously, the voter does not engage in a multi-step protocol with the device, so there is little scope for any "social engineering"-style attacks. It is worth noting that in Prêt à Voter, the analogue of the "cut-and-choose" element of the Chaum or Neff schemes lies in the random auditing of the ballot forms, that is, before any voter involvement. The authority commits to the cryptographic material on the ballot forms ahead of the election, and a random selection of the forms are checked by the auditors for well-formedness. Assuming that they pass the checks, audited forms are destroyed. Any forms that fail the checks would be retained for forensic purposes.

In addition to checks performed by auditors, voters are also able to carry out random checks on Prêt à Voter ballot forms. In this way, trust can be arranged, ultimately, to reside in the voters themselves. Care has to be taken, however, that providing such features does not introduce other vulnerabilities, such as the possibility of checking ballot forms that have already been used to cast votes.

### 6.2.3 Denial of Service

Although the device does not create receipts in Prêt à Voter, it could still duplicate or corrupt them, causing a denial of service attack. Duplicated receipts could be deleted if the onion values and candidate choices are identical, but this is more problematic if the latter turn out to be different. Furthermore, in the case of erroneous onion values, voters would not be able to confirm the correct posting of their receipts. Selective denial of service attacks by colluding tellers is also possible. As with the Chaum and Neff schemes, recovery and error-handling strategies are essential.

A possible enhancement of Prêt à Voter is to replace the decryption with re-encryption in the mix process [165]. This has a number of advantages, one being that recovery from a denial of service attack is much easier. There are a number of reasons for this:

- The mix tellers do not need secret keys, they simply re-randomise the encryption. A failed mix teller can therefore simply be replaced without having to surgically extract keys.
- The mix and audit can be independently re-run. With deterministic decryption mixes, the links for auditing cannot be independently selected on the re-run of the mix without compromising secrecy.

The use of threshold encryption schemes would help foil denial of service attacks by ensuring that the failure of a proportion of the decryption tellers could be tolerated by the system.

It has been suggested that a *voter-verifiable paper audit trail* (VVPAT) [119]-style mechanism should be incorporated in Prêt à Voter [146]. In other words, as the device scans the voter's receipt, it generates an extra copy. Once this copy has been verified by the voter, and possibly an official, it is entered into a sealed audit box. This provides a physical back-up of receipts cast, should recovery mechanisms need to be invoked. It should be noted that this is quite different to a conventional VVPAT of unencrypted ballot slips, which suffers from a number of problems concerning privacy. For example, receipts could be linked to voters by comparing the order in which votes are cast with that of voters entering the booth. Furthermore, a mismatch between the voter's choice and the paper audit record can be difficult to resolve without compromising the voter's privacy. These problems do not occur when encrypted receipts are recorded. Another advantage is that, voter verification of the ballot receipt record could be reinforced by monitors. Any discrepancies can be resolved without compromising the voter's privacy. Henceforth, such a mechanism on encrypted receipts will be referred to as a *verified paper audit trail* (VPAT).

### 6.2.4 Discarded Receipts

Like the Chaum and Neff schemes, carelessly discarded receipts could be a problem in Prêt à Voter, as this could indicate which receipts will not be checked on the Web bulletin board [92]. Malicious colluding parties could then delete or alter the corresponding ballots, possibly also injecting votes of their own choosing into the system, with a lower probability of being caught.

Aside from voter education [92], a possible mitigation is, again, to implement a VPAT mechanism, as described above. To catch any discrepancies, independent observers could check the correspondence between the VPAT and the contents of the bulletin board. The added advantage is that less reliance need be placed on the voter's diligence in checking that their receipts have been correctly posted.

### 6.2.5 Invalid Digital Signatures

In the Chaum scheme, digital signatures act as a counter-measure against faked receipts being used to discredit election integrity. However, false signatures could be used to discredit voters, leaving them without a way to prove a dishonest system [92]. As a countermeasure, voters should be provided with a means for checking the digital signature, preferably by independent organisations, such as the U.K. Electoral Commission.

Similar measures could be utilised for Prêt à Voter. In addition, given the possibility of casting encrypted receipts in the presence of officials and other observers, physical authentication mechanisms, such as franking, could be employed. However, precautions against forged stamps would be desirable.

### 6.2.6 Insecure Web Bulletin Board

Like the Chaum and Neff schemes, Prêt à Voter also relies on a secure Web bulletin board, which, aside from it having read-only access, is otherwise unspecified. In a possible scenario, the bulletin board arranges for the voter to see a correct record of her ballot receipt, which, in collusion with the mix-net, has been deleted or altered. As a result the voter could mistakenly believe that her vote has been accurately counted [92].

Note that suggested mitigations, such as robust data storage and allowing only authorised write access to the Web bulletin board [92] are still vulnerable to corruption. The challenge is provide a trusted path from the bulletin board to the voter, and is the subject of ongoing research in the cryptographic voting community.

## 6.3 Further Vulnerabilities

In this section, other possible vulnerabilities, not identified in [92], are discussed. Where possible, appropriate mitigation strategies are suggested.

### 6.3.1 Doll Matching Attack

Recall that the Chaum scheme requires the voter to check that the pair of "dolls", which encode the information necessary for decrypting receipts and are printed by the device on the two layers, match. Failure to do this could allow the device to construct fake receipts without detection. It might appear difficult to produce a fake "doll" that alters the vote value, whilst differing from the real "doll" in a way that is, visually, almost imperceptible. This would be analogous to finding (approximate) collisions of a cryptographic hash function. Nevertheless, it should be considered a potential vulnerability.

A possible counter-measure would be to ensure that visual matching of the "dolls" is as easy to perform as possible. For example, the "dolls" might be encoded in vertically aligned bar codes on either side of the central perforation. Any mismatch would then show up as a misalignment of the bars.

### 6.3.2 Undermining Public Confidence in the Secrecy of Encrypted Receipts

Another potential attack against schemes employing encrypted receipts, not identified in [92], is as follows. The voter is falsely persuaded that there is some way to extract her vote from the encrypted receipt. If a sufficient number of voters were convinced by such a claim, and thus influenced to alter their vote, it may be possible to undermine the outcome of the election. For instance, a coercer might urge voters to submit their receipts, claiming that the "correctness" of the choice would be checked. Some reward, such as entry into a lottery, could be offered for receipts that passed the supposed checks. Countering such a psychological attack, other than by voter education, could be difficult.

## 6.4 Prêt à Voter Specific Vulnerabilities

The analysis of Prêt à Voter revealed other possible vulnerabilities that are specific to the scheme. They are discussed in the following, together with possible mitigations.

### 6.4.1 Chain Voting

Chain-voting is a well known attack that can be effective against some conventional paper ballot schemes. In this attack, the coercer somehow obtains an unused ballot form and marks his preferred candidate. A voter is told that they will be rewarded if they use the marked form, and return with a fresh, unmarked form. This can then be marked again and passed to the next voter.

Election systems in which the ballot forms are a controlled resource are particularly vulnerable. On registration, voters are given a ballot form which they are observed to use before exiting the polling station. Even though she might be under duress to cast the ballot form marked by the coercer and to retain the fresh one, a wily voter might still succeed and escape detection. Arguably, the procedures make it harder for the coercer to initialise the attack, although, again, with determination, a way could be found. A coercer could, for example, bribe an official, or initiate the attack by placing a fake form in the ballot box, and smuggling out an unused one.

A possible counter-measure is to make ballot forms freely available in the polling stations, as, for example, in French elections. Voter identity is checked prior to casting a vote, rather than at the time of collecting a ballot form. This increases the uncertainty that a marked form was actually used to cast a vote, and, hence, motivation for the attack is undermined.

Neither the Chaum nor the Neff schemes, in which the ballot forms and receipts are generated on demand in the booth, are open to chain voting. However, it is a potential threat in Prêt à Voter, and the counter-measure suggested above fails, as cast receipts are posted to a publicly-verifiable bulletin board.

### 6.4.2 Mitigation

In Prêt à Voter, the problem is that the coercer may be able to induce a voter to use a ballot form that he has seen, and hence knows the association between the onion and candidate list. Although, various counter-measures can be devised, this is further evidence for the more general point that the information on ballot forms has to be carefully protected. I return to this issue later.

Concentrating on the chain voting threat for the present, a first observation is that the voter does not, in fact, need to know the onion value in order to make their selection on the form. This suggests a mechanism to ensure that the onion value is only revealed after the ballot has been cast. A further observation is that, in a scheme using encrypted ballot receipts, only marking the vote choice requires the privacy of a booth. The actual casting of the ballot receipt could be performed in the presence of an official.

Together, these two observations suggest the following counter-measure. The onion is concealed with a "scratch strip", similar to the one commonly used in lottery tickets. An example is shown in Figure 6.3. A possible procedure would then be for the voter to register and collect a fresh ballot form, with scratch strip intact. The voter then goes to the booth and marks her vote, then detaches and destroys the left-hand strip. Next, she exits the booth and takes her receipt to an official who checks her identity, scores off her name from the electoral list, and checks that the scratch strip is intact. The strip could then be removed to reveal the onion, and the receipt recorded as previously described in Chapter 4.

Steps would need to be taken to ensure that the scratch strips cannot be scanned with some device to read the concealed onions. This has reportedly been done using the laser photo-acoustic effect [63]. A laser beam is used to line-scan the strip while a microphone picks up the acoustic waves caused by differential absorption of the light. By adjusting the effect to the appropriate depth, the printing below the covering strip can, apparently, be read. Obtaining and operating the necessary equipment would, however, require expense and technical know-how on the part of the attacker. Nevertheless, it should be considered a possible threat.

In [146], supervised voting, that is, in the presence of an official, was suggested as a countermeasure against double voting. The above suggests another reason why this might be desirable.

A rather different approach is to implement on-demand creation of ballot forms, which averts chain voting and certain chain of custody issues with ballot form information. However, the cost is having to introduce the "cut-and-choose" element and post auditing into the protocol. The trade-offs involved in this are investigated in [159].

### 6.4.3 Authority knowledge

In the current version of Prêt à Voter, the authority has knowledge of ballot form information, that is, the cryptographic material used to generate the onions, and, in particular, the association between the onion values and corresponding candidate lists. This could be problematic, as the authority has to be trusted not to disclose this information. Even if the authority is entirely trustworthy, there is always a danger of this information being leaked during storage or distribution of the ballot forms. As mentioned earlier, the possibility of kleptographic channel attacks have been identified in [65]. They can occur if information is leaked through a trapdoor built into system code, and are difficult to detect. One way to prevent this is for no single party to have access to sensitive information.

A possible solution would be for the ballot form material to be constructed in a distributed fashion, so as to ensure that no single entity knows the association of onions and candidate lists. As noted previously, up until the time of casting a vote, it is not necessary for the voter to know the onion value. One could devise a similar scheme in which the onion and corresponding candidate list are never exposed simultaneously. The challenge is to guarantee that the candidate list and the onion that is later shown to the voter are correctly linked. Ideally, this should be done without having to place trust in any devices or individuals. This is currently ensured by the fact that the authority commits to the association in print before random auditing.

The following is an outline of a simple scheme for distributed construction of scratch card style ballot forms. It is based on onions encrypted using El Gamal, or a similar randomised encryption algorithm. The ballot form material is generated by a number of ballot clerks. The first clerk generates a large quantity of onions, which then enter a re-encryption pre-mix involving the other clerks. The last clerk collects the resulting permuted, random onions and, for each one, produces two re-encryptions. These paired onions are now printed onto ballot forms, one at the bottom of the left-hand column, the other on the bottom of the right-hand column. A proto-ballot form is shown below:

These two onions should correspond to the same candidate list. The right-hand onion is now concealed with a scratch strip, as shown below.

Both the onion pairs and ballot forms are now shuffled, as a precaution against possible collusion between the ballot clerks. The (shuffled) onion pairs and ballot forms are passed to a final clerk,

| | |
|---|---|
| | |
| | |
| | |
| | |
| | 7rJ94K |
| jH89Gq | |

Figure 6.1: Distributed Ballot Form Creation - Step 1

| | |
|---|---|
| | |
| | |
| | |
| | |
| | ■ |
| jH89Gq | |

Figure 6.2: Distributed Ballot Form Creation - Step 2

who sends the visible, left-hand onions to the tellers. The tellers return the corresponding candidate lists, which are then printed in the left-hand column of each form and the corresponding onion is removed. This results in ballot forms, similar to the example given in Figure 4.3, but with the onion concealed. These can checked for well-formedness as described previously.

| | |
|---|---|
| Nihilist | |
| Buddhist | |
| Anarchist | |
| Sophist | |
| | ■ |

Figure 6.3: Ballot Form With Concealed Onion

Note that no single entity now knows the association of onion and scratch strip. Strictly speaking, the last two clerks acting in collusion could form the association, but the scheme can be elaborated to raise the collusion threshold. Although these two clerks in collusion might know the onion/candidate list association, however, they cannot prove this knowledge to a third party, as they do not know the necessary cryptographic primitives.

Distributed generation of the ballot forms could be done in such a way as to ensure that only a threshold set of entities are able to recover the candidate list and onion associations. A cryptographic analogue of the above process is described in [165].

### 6.4.4 Enforcing the Destruction of the Left-hand Strips

After the voter has marked her choice on the ballot form, it is important that the left-hand strip be destroyed. Otherwise, it could be used to prove a vote to a third party, which, clearly, lays the system open to coercion.

Several mitigations against this are possible. The voter could be required to destroy the left-hand strip in the presence of an official, preferably in some mechanical shredding device. This could be done at the time of casting the ballot form, as suggested previously. However, care would have to be taken to ensure that the official is not able to record the associations between the right- and left-hand strips. In other words, the ballot receipts and candidate lists.

Another possibility, is to install devices in the booths that would automatically separate and destroy the left-hand strips, and pass the receipts into an optical reader. The voter's interaction would be made easier, but at the cost of placing trust in devices, which runs counter to the philosophy of current voter-verifiable schemes. Alternatively, as suggested previously, "decoy" left-hand strips could be made freely available in the booths, so it would be difficult for the voter to provide convincing proof of a vote.

### 6.4.5 Confusion of Teller Modes

As previously mentioned, the tellers perform an anonymising decryption mix on the receipts posted to the Web bulletin board. However, they also have a role in checking the construction of ballot forms, both by auditors and voters [36]. During an audit, the onions are sent to the tellers, who return the corresponding cryptographic primitives necessary to re-compute the onion values and candidate offsets. The auditors then check that they are correct. In voter checking, the onion value is sent to the tellers, who return the associated candidate ordering.

The checked forms should then be discarded, as the associations between the onions and candidate orderings have been revealed. If the forms were to be re-used, there could be a threat to ballot secrecy. Conversely, it should not be possible to run a check on a form that has been used to cast a vote. To mitigate this, ballot forms could be checked by voters in the presence of an official, who then ensures that used forms are discarded. It could be arranged for forms to be invalidated once used, for example, using a "scratch strip" mechanism, similar to the one shown in Figure 6.3. An authentication code, required at the start checking, could be overprinted on the scratch strip concealing the onion value. Revealing the onion would entail removing the scratch strip and the code along with it, ensuring that the form could not be reused later.

## 6.5 Discussion and Summary

In this chapter, a systems-based analysis of Prêt à Voter has been discussed. The aim was to assess the security of the scheme, taking into account interaction of entities, such as hardware, software, the authority, tellers and voters, with the protocol. Certain vulnerabilities not mentioned in [92] were identified, thus extending the analysis of Karlof et al. The applicability of these vulnerabilities to the Prêt à Voter scheme has been discussed, and where possible, various counter-measures suggested.

Prêt à Voter has proved to be remarkably resilient to many vulnerabilities which stem from voter interaction with devices and generation of entropy at the time of voting. This is perhaps not surprising, as Prêt à Voter does not feature a complex exchange between the voter and booth device. Furthermore, as the ballot forms are generated before voting begins, any necessary entropy is pre-determined. However, Prêt à Voter is potentially prey to chain voting attacks, which do not apply to schemes in which the cryptographic material is generated on demand. In fact, as has been discussed, this attack is particularly virulent in the context of voter-verifiable schemes with pre-prepared ballot forms and Web bulletin boards. A possible counter-measure is to conceal the onion on a ballot form with a scratch strip until a vote is actually cast. Others are currently being evaluated as future enhancements to the scheme.

Although of it has proved extremely useful, the approach of Karlof et al could not be regarded as methodical. Thus, it may not represent an exhaustive identification of all the system-based threats in the Chaum and Neff schemes or, as described above, in Prêt à Voter. Given the open-ended nature of systems, complete coverage for such an analysis could never, perhaps, be guaranteed. However, this analysis constitutes a valuable first step towards a more systematic analysis technique for voting systems. It has also indicated a "taxonomy of attacks", that is, classification into subliminal channels, "social engineering" threats, etc. It is possible that, together with a design-level information flow analysis, similar to the one presented in the previous chapter, a more rigorous form of analysis could be devised. This would be an interesting subject for future research.

It has been made clear that voting schemes, as with any secure system, require great care in the design and evaluation, not only of the cryptographic core, but also of the surrounding system. Analysis from a system perspective has also provided valuable insight into the way forward for Prêt à Voter, and some of the planned future enhancements have been mentioned. It has also underlined the need for adequate error-handling and recovery strategies, and that a VPAT-style mechanism would be highly desirable.

Provided that an adequate security analysis is performed during the design and evaluation phases, there is every reason to suppose that cryptographic schemes, such as Prêt à Voter, can provide trustworthy, verifiable elections.

# Chapter 7

# Coercion-resistance and Receipt-freeness

An essential requirement of voting systems is coercion-resistance, as it protects the voter's right to vote according to her free will. It also has an impact on other requirements such as ballot secrecy, legitimacy and eligibility. Loss of ballot secrecy lays a scheme open to coercion, as the value of a coerced vote is no longer confidential. Legitimacy may be undermined as the coercer can potentially vote more than once by forcing or bribing several voters to vote in a certain way. Furthermore, he may not be an eligible voter. Election integrity could also be at risk, as coerced votes may not reflect the electorate's true choice.

Although the terms "coercion-resistance" and "receipt-freeness" are sometimes used interchangeably, coercion-resistance has been formalised as a stronger property [91], [47]. "Receipt-freeness", usually means that a voter is unable to construct a proof of her vote, whereas coercion-resistance could be defined as the inability of the coercer to distinguish between an instance when the voter votes $V_i$, and when the voter votes $V_j$, where $V_i$ is a vote supplied by the coercer, possibly null or a random value, and $V_j$ the voter's choice. Both properties deter an outsider from trying to persuade the voter to vote in a certain way. One way of drawing out the distinction between these two properties is to think of the coercer as being "passive" in the case of receipt-freeness, and "active" in the case of coercion-resistance. Juels et al [91] define coercion-resistance as resistance to forced abstention, randomisation and simulation attacks in addition to the inability to prove the value of a vote. The various definitions are discussed in more detail shortly.

The first proposal for a "receipt-free" voting system is attributed to work by Beneloh et al [21]. Several other schemes have ensued, for example [136], [168], [136], [18], [82], but this is by no means an exhaustive list. In [91], coercion-resistance is achieved by issuing voters with "credentials", which could be secret/public key pairs. The voter has the option of casting a vote encrypted either with a valid or, alternatively, a "fake" key, the coercer being unable to distinguish between the two encryptions. Similarly, in [38], voters have the option of voting with either a valid "capability" or a random string. During the tallying phase, only the valid votes, that is, the votes encrypted with legitimate keys or cast using a valid capability, are counted. This approach is described later in the chapter.

As demonstrated in this chapter, another way to express indistinguishability between "true" and "false" votes, is by using the opacity property [25], [26], [27] Recalling the informal definition of opacity, a predicate, $\phi$, is opaque if for every run in which $\phi$ is true, there exists an indistinguishable run in which it is false. For example, $\phi$ could be defined as being true of a trace of the system if it contains the valid encryption of the voter's "true" vote. To satisfy opacity, an observer should be unable to distinguish between valid and invalid encryptions of votes that may be cast by the voter. Note that the "observer" in this case is anyone eavesdropping on an execution of the voting protocol, other than the voter and possibly the authority. Clearly, the voter will know how she voted, and the authority may have access to the voter's keys. Note, however, that authority knowledge of ballot information has been identified as a potential vulnerability in Prêt à Voter, and a scheme for distributed ballot creation has been proposed [165]. Likewise, in a remote implementation of Prêt à Voter [38], methods are being investigated for construction and distribution of capabilities to minimise trust in the authority. The difficulty, however, is proving the validity of a capability only to the designated voter. This is currently under investigation.

In this chapter, two voting schemes are analysed for their coercion-resistance, namely Prêt à Voter [36] and the FOO [59] scheme. As discussed in previous chapters, Prêt à Voter is fairly resistant to coercion [163], whereas the FOO scheme is vulnerable to this type of attack. Even without weaknesses in the protocol, coercion-resistance in a remote voting scheme, such as FOO, cannot be guaranteed. The the attacker could be physically present at the time of casting a vote, and the voter could be forced to vote as instructed. The voter could also hand over her voting keys to the attacker, either willingly of under threat. I discuss this further in the next section. As discussed previously, another complication with the FOO scheme is that communication between the voter and the system must be anonymous, which can also be difficult to implement.

In each case, the scheme is assessed for coercion-resistance, taking into consideration the different possible capabilities of an attacker. The aim is to determine whether the requirements of each scheme for coercion-resistance can be expressed in terms of a suitably defined form of opacity. As seen shortly, this is indeed the case, at least informally. Following the analysis, receipt-freeness is formally cast as opacity [138]. The possible way in which this definition could be extended to capture coercion-resistance is discussed.

I also investigate possible relationships between the opacity property and some related cryptographic primitives, such as designated verifier proofs [88] and deniable encryption [30]. They are each described in the relevant sections. A possible relationship between anonymity and coercion-resistance is also considered. I begin by discussing coercion-resistance in further detail.

## 7.1 Coercion-resistance

Coercion in voting systems can occur if a voter is persuaded to vote in a certain way by a third party. As discussed in the previous chapter, chain-voting could be regarded as a form of coercion. In chain-voting, an outsider obtains an unused ballot form and marks it with his vote selection. He then either pays or forces a voter to cast it at the polling station. If the voter returns with a fresh form, the attack can be repeated.

As mentioned above, coercion-resistance is sometimes referred to as receipt-freeness. Both properties demand that the voter has no means of proving which way she voted to a third party, although coercion-resistance accounts for possible intervention in the voting protocol on the part of the coercer. Note that the term receipt-freeness can be misleading. In some schemes, such as FOO, the voter does not obtain a physical receipt, but may still be able to construct a proof of her vote. In the FOO scheme, the voter submits her vote, which has been committed with a random factor. She later reveals the value of her vote using this random factor when checking a publicly-accessible bulletin board. This allows the voter to verify that her vote has been correctly recorded by the system, but the loophole is that the voter merely has to supply the coercer with the random factor to prove her vote. Alternatively, the voter could use a random factor provided by the coercer. In either case, coercion-resistance fails.

In some schemes the voter does obtain a physical receipt, but it is constructed in such a way that it does not provide readily verifiable proof of a vote to a third party. Examples of voting systems with this feature are Prêt à Voter and the Chaum [35] and Neff [131],[132], [133] schemes. As described in the previous chapter, the coercion-resistance of these schemes has been analysed from the systems perspective [92], [163], that is, by examining interactions between the various components of each system.

In both the Chaum and Neff schemes, ballot receipts are generated on demand, and receipt-preparation involves random choices made at the time of casting a vote. As described in [92], a voter in the Neff scheme may be able to prove the value of her vote by choosing the randomness, perhaps pre-arranged with the coercer, in the construction of her receipt. Note, however, that this may no longer be the true of the scheme in its current implementation [5]. Although essentially deterministic, the Chaum scheme is based on visual cryptography. Individual votes could be identified if there is more than one valid representation of the vote image, and the choice of image is pre-arranged between a corrupt third party and voting device [92].

Prêt à Voter ballot forms are printed ahead of time, and the scheme does not require random input from either the voter or the voting device. Hence, the subliminal channel attacks as described above, are not a problem. However, the possibility of kleptographic channel attacks has been identified [65]. In creating the ballot forms, the authority would select seed values in such a way as to encode the ordering of the candidate lists in the onion values, and so that a certain keyed hash applied to the onion value would reveal the corresponding candidate order. If the key is shared with a colluding third party, this could lay the scheme open to coercion. Chain-voting, which could be regarded as a form of coercion, is also possible. Both attacks have been discussed in Chapter 6, along with possible countermeasures. For further details, the reader is referred to [92], [163] and [165].

Note that coercion-resistance may also fail if the coercer knows exactly how many votes his chosen candidate will get, and can work out whether or not the target voter voted in a certain way. Juels et al observe that coercion-resistance depends to an extent on the coercer's knowledge of how the "honest" voters will vote. In reality, the coercer may have an idea of percentage votes, but not exactly how many [91].

In this chapter the coercion-resistance of Prêt à Voter is re-investigated, but from a different angle, that is, in terms of opacity. I next review some related work.

## 7.2 Related work

As mentioned earlier, formal definitions have been proposed for coercion-resistance in [91] and [46]. Juels et al verify their definition of coercion-resistance using cryptographic proof techniques [91]. This is based on the computational infeasibility of the attacker guessing whether the voter evades coercion, hands over a "fake" key, or submits to coercion, that is, hands the "true" key over to the coercer. This definition of coercion-resistance includes resistance against randomisation, forced abstention and simulation attacks, in addition to the inability of a voter to prove her vote. In a randomisation attack, the voter is coerced into to casting a randomly composed ballot, effectively nullifying her vote with a large probability. The voter is also under duress to cast the ballot provided by the coercer, if she is unable to determine the value of the vote. In a forced abstention attack, the voter is bribed or forced not to vote at all. As an example of a simulation attack, the coercer casts a vote using the keying material issued to the target voter. As described in the analysis to follow, this can occur in the FOO scheme. During the course of the analysis, I discuss this notion of "full" coercion-resistance in relation to Prêt à Voter.

Delaune et al provide definitions for both receipt-freeness and coercion-resistance using the applied pi-calculus [8]. They show that receipt-freeness implies privacy, and that coercion-resistance implies receipt-freeness. To model coercion-resistance, it is assumed that the voter, $V$ interacts with the coercer, $C$, whereas with receipt-freeness, there is no interaction between $V$ and $C$ during the protocol, but $V$ can later prove her vote to $C$.

As mentioned previously, absolute coercion-resistance is probably impossible to achieve in a remote voting system. It is difficult to prevent the physical presence of a coercer at the time of a vote is cast, and there needs to be an opportunity for the voter to cast her vote unobserved.

Clarkson et al address this problem in a remote implementation of Prêt à Voter [38]. The voter is issued a "capability", which is essentially an encrypted proof of validity. The voter can choose to submit a vote using this capability, or some random string. The idea is that the coercer should not be able to distinguish between them. If under duress, the voter could use the "fake" capability, perhaps submitting a vote of her own choice using the valid capability at a later time. In the case of vote-buying, the voter would not be able prove how she voted to a third party, as it would be uncertain whether she used the valid or the "fake" capability. In the tallying phase, the invalid votes would be discarded. This is similar in principle to work by Juels et al [91], mentioned earlier, in which the voter has the choice of using either a valid or "fake" credential.

In [46], Delaune et al describe an attack similar to randomisation [91], but in addition to rendering votes as invalid, enables the coercer to identify voters who do not comply with his wishes. They show that their definition for receipt-freeness, as given previously, is secure against this attack. Work is reported to be in progress on a receipt-free protocol which utilises valid and "fake" keys [46]. The scheme works in a similar way to that in [38]. Delaune et al [46] observe that coercion-freeness is related to secrecy in that only the voter knows how she voted. However, it is necessarily a stronger property to account for the possibility that the voter may cooperate with the coercer.

Another possible way to express coercion-resistance, based on indistinguishability between two events, is in terms of a suitably defined form of opacity. As will be seen shortly, the extreme flexibility of the opacity property is also an advantage as it can capture different forms of coercion-resistance, such as resistance to forced abstention [91].

By way of an illustration, I now analyse the requirements for coercion-resistance in Prêt à Voter, and express them as a variation of opacity. This analysis could be regarded as a combination of the systems-based and information flow analyses carried out in Chapters 6 and 5. While the focus is on information flow, interactions between the different components of the system are also taken into account. As noted previously, this may constitute a more rigorous approach to the one taken in [92] and [163]. The following analysis [138] may exemplify the latter, and provide insight for future work on the subject.

## 7.3 Coercion-resistance in Prêt à Voter - a New Angle

In this section, the requirements for coercion-resistance in Prêt à Voter are informally expressed as variations of the opacity property. Examining the scheme, there appear to be four cases to consider, all of which depend on the power of the coercer. They are discussed below. For a description of Prêt à Voter, the reader is referred the overview given previously in Chapter 4, or to [36] for full details.

- The coercer is unable to obtain any unused ballot forms and can only observe the Web bulletin board.

The coercer is only able to scan the published list of encrypted receipts, and also the list of decrypted votes. As he has not seen any ballot forms, he does not know the associations between the onion values and corresponding candidate lists. Even if he managed to persuade a voter to make a particular choice, the published lists do not provide any evidence of how she voted. As discussed previously, this is true as long as destruction of the left-hand strips is enforced, or "decoy" left-hand strips are made freely available in the booth. Otherwise, the voter may be able to prove her vote.

Recalling the informal definition of opacity, the predicate, $\phi$ is opaque, if for every run of the system in which $\phi$ true, there is an observationally equivalent run in which it is false. To expressing the above in terms of opacity, the $\phi$ is first defined being true of a trace if it contains the value of the voter's true vote. Opacity holds over the voting system if, for every event in which the voter casts her true vote, there is another such event corresponding to the same voter, but with a different vote, and the coercer should not be able distinguish between the two events. This accounts for the possibility that one of the votes was chosen by the coercer.

As will become clearer shortly, to capture coercion-resistance in Prêt à Voter, it is, in fact, necessary for $\phi$ to be defined as being true of a trace of the system if it contains the association between the onion and corresponding candidate list denoting the voter's true choice. In addition, $\phi$ should be opaque with respect to an appropriately defined *obs* function. Note that the coercer could be any observer, except the voter herself.

- The coercer can obtain an unused ballot form and also observe the Web bulletin board.

In this case, the coercer can note the onion value on the ballot form, make his vote selection, and threaten or bribe a voter to cast it at a polling station. He can then find out whether or not the voter followed his instructions by scanning the published list of encrypted receipts.

The second case illustrates the way in which a chain-voting attack could be initiated in Prêt à Voter, that is, by an outsider getting hold of a fresh ballot form. As suggested in Chapter 6, a possible counter-measure is to conceal the onion values, for example, with scratch strips, until votes are actually cast [163]. In [165] Ryan et al propose a variant of this idea, whereby the candidate list is concealed until the time of vote casting.

With the scratch strip mechanism in place, the coercer would not be able to associate onion values with corresponding candidate lists, and hence, to identify any one receipt from the list published on the bulletin board. Here, $\phi$ could be defined as being true of a trace if it contains the association between the onion and corresponding candidate list denoting the voter's true choice. For opacity to hold, the coercer should not be able to distinguish between an event in which the voter casts her true vote, and one in which she (possibly) casts the vote supplied by the coercer. I examine this in more detail shortly. In a physical sense, the scratch strips serve to render the value of the votes opaque.

I next re-examine the scheme with concealed onion values.

- The coercer can obtain an unused ballot form and also observe the Web bulletin board. However, the onion value has been concealed with a scratch strip. The coercer cannot determine the onion value without removing the scratch strip and invalidating the ballot form.

  Recall that in the booth, the voter marks her choice on a ballot form, and detaches and destroys the left-hand strip. She next presents her receipt to an official who checks that the scratch strip is intact. The voter now removes the scratch strip and casts her vote. This rules out the possibility of an outsider having seen and noted the association between the onion and corresponding candidate, which as before, is opaque to the observer.

- A further case is where the coercer can obtain unused ballot forms, observe the Web bulletin board, and is also in collusion with a corrupt authority who has complete knowledge of the ballot form information.

  The target voter follows the protocol described above. However, a malicious device records the onion value, and, in collusion with the corrupt official, links it to the voter's identity. Once in possession of the voter's receipt, the coercer presents the onion value to the authority, who returns the associated candidate ordering and the voter's identity. He can then check the vote value.

  Hence, even with the onion values concealed by scratch strips, the coercer can determine the value of a vote cast by the target voter if the necessary information is leaked by the authority. He does not need to obtain any unmarked ballot forms, or observe the Web bulletin board.

  To defeat coercion, an alternative to the scratch strip mechanism is for the ballot material to be generated in such a way that no single party has complete knowledge of the association between the onion values and corresponding candidate offsets. In this case, the requirement for coercion-resistance could still be expressed with the opacity statement given above. A possible scheme for distributed ballot creation in Prêt à Voter is proposed in [165].

The above analysis illustrates that one possible way to increase coercion-resistance in Prêt à Voter is to prevent chain-voting attacks. The use of the scratch strip mechanism suggested in [163] has been considered, and the importance of avoiding single authority knowledge of ballot form information has been highlighted.

Note that, "coercion-resistance" as used above, includes all the criteria set out by Juels et al in [91], that is, resistance to forced abstention, randomisation and simulation attacks, as well as the inability of a voter to prove her vote to a third party. The first is true unless the coercer can obtain evidence that the target voter actually cast a vote. This would be difficult in a supervised scheme, such as Prêt à Voter, unless the coercer is able to closely monitor the procedures inside a booth. As Prêt à Voter ballot forms are prepared ahead of time, and a vote is indicated only by a mark against the chosen candidate, randomisation attacks should not be a problem. The coercer would not be able to supply the voter with a randomly constructed ballot form, unless he was in collusion with the authority. Note that this attack is distinct from one utilising random subliminal channels, as described in [92], where either the voter or device provide carefully chosen random inputs to create a ballot receipt. Chain-voting could be regarded as a form of simulation, since the coercer effectively votes under another identity. However, as before, without knowing the associations between onion values and corresponding candidate lists, the coercer would have no proof his vote was cast. Note also that "full" coercion-resistance [91] holds as long as the coercer is not in league with the authority.

From the analysis, it can be seen that there are no opportunities for coercion in Prêt à Voter via subliminal channels as identified in [92]. However, finding the most effective and viable defences against chain-voting attacks is the subject of ongoing research.

In the next section, the analysis of Kremer et al in [94] is extended with an analysis of coercion-resistance in the FOO scheme [138].

## 7.4 Coercion-resistance in FOO - an Analysis

As previously mentioned, the FOO scheme has been analysed for several security requirements, such as ballot secrecy, legitimacy and fairness in [59] and [94]. Both papers acknowledge, somewhat indirectly, that the scheme does not satisfy receipt-freeness but do not discuss this problem in any detail. To the best of my knowledge, this is the first analysis of coercion-resistance in the FOO scheme. For a description of the FOO scheme, the reader is referred the overview given previously in Chapter 4, or to [59], [94] for further details.

Studying the protocol, it appears that there are three cases to consider, each reflecting different powers of the coercer. They are as follows.

- The coercer can only observe the bulletin board, and does not know the value of the random factor used by the target voter to commit her vote.

  Recall that the voter, $V$'s commitment to her vote, $c = \xi(v, k)$, which has been signed by the Administrator, is initially published alongside an index number, $l$. The voter verifies her vote, and then sends $l$ and $k$, the random factor used to commit her vote, to the Collector, who obtains and finally publishes the value of $v$. Without $k$, the coercer would not be able to construct a proof of $V$'s vote.

If $\phi$ is defined as being true of a trace if it contains the value of $V$'s true vote, then the system satisfies opacity with respect to $\phi$ and $obs$ if the coercer cannot distinguish between two events corresponding to the same voter, one in which she casts her true vote, and one in which she casts a vote under the coercer's influence. As will be demonstrated shortly, for coercion-resistance in FOO, $\phi$ should be defined as being true of a trace if it contains the value of the random factor used by $V$ to commit her vote, which should be opaque to the coercer.

Note that in the original paper, the actual set of values from which the voter makes her selection is unspecified. It is conceivable that the coercer could supply the voter with some unique or random value, in which case, just by looking at the final list of decrypted votes, he would be able to determine whether or not his instructions were carried out. However, if the value of the vote is not unique, then without the voter's cooperation, either forced or willing, the scheme is, in this case, coercion-resistant.

The next two cases show how the voter could be coerced.

- The coercer computes a blinded, committed vote. The target voter, either willingly or under duress, digitally signs and sends it to the administrator, along with her ID. Alternatively, the voter could supply the coercer with her signing keys. When the encrypted vote has been signed and returned by the administrator, the coercer then performs the remaining steps in the protocol.

  This attack demonstrates that coercion-resistance is difficult to guarantee in a remote system if the coercer manages to obtain the voter's keys and evades authentication. Having computed the ballot, and with knowledge of the required cryptographic values, the coercer can complete the protocol for his vote choice using the voter's digital signature. Note that this is an example of a simulation attack [91].

- The coercer supplies the target voter with a vote. The voter computes a blinded, committed vote, then follows the steps for registration and voting. When the list of encrypted votes is published, the voter supplies the coercer with $k$, the random factor used to commit the vote. The coercer then checks that the supplied vote has been correctly transmitted, and if satisfied, either completes the final steps or instructs the voter to do so.

  Similar to the case above, this again illustrates the difficulty of achieving coercion-resistance in a remote system, where a third party can observe the voting process. It also shows a weakness in the protocol in that once the coercer obtains the random factor, $k$, the proof of a vote is irrefutable. I discuss this further shortly.

It appears that, in the FOO scheme, coercion-resistance holds for a "passive" coercer, that is, one who only observes the published lists of encrypted and decrypted votes. In this case the value of the voter's vote, or more precisely, the random factor used to commit the vote, is opaque to the coercer. As discussed previously, the "coercer" could be any observer other than the voter.

The remaining two cases in analysis above show that the FOO scheme is weak in terms of coercion-resistance. This is partly due to the problem of preventing third party intervention during remote voting, as shown in the second case, and partly due to the design of the protocol, as shown in the third case. In [135], Okamoto proposes a voting scheme similar to FOO, in which trapdoor bit-commitment is used to enable multiple possible decryptions of a committed vote. However, he later showed that the scheme also required physical assumptions, such as untappable channels or a voting booth to satisfy receipt-freeness [136]. Further treatment of this is beyond the present scope, and the reader is referred to [135] and [136] for details.

From the analysis of Prêt à Voter and FOO, it appears that variations of the opacity property can be used for succinct expressions of coercion-resistance. By the definition of opacity, the observer should not be able to distinguish between two runs of the system, one in which the predicate, $\phi$, is true, and another in which it is false. This relates closely to recent models of coercion-resistance where a coercer is unable to establish whether the target voter used a "fake" or valid encryption of her vote [91], [38]. It also takes care of the possibility of the coercer confirming a putative cryptographic value, that is, succeeding in a guessing attack.

Furthermore, by altering the definition of the predicate, $\phi$, and the observer's view, opacity adapts readily to the requirements of two markedly different voting protocols, in this case Prêt à Voter and the FOO scheme. It would be interesting to apply a similar analysis to other voting schemes as a future extension to this work.

So far, informal opacity statements have been used to express particular requirements for coercion-resistance. In the next section, the intuitive statements are consolidated by a formal statement of general receipt-freeness in terms of opacity. By defining $\phi$ and the observation function, $obs$, as appropriate, this could potentially be applied in a variety of different situations, and in a range of different systems, As will be explained shortly, a formal definition of coercion-resistance as opacity is rather more complicated.

## 7.5 Receipt-freeness Cast as Opacity

In carrying out the analysis of coercion-resistance in Prêt à Voter and the FOO scheme, a "threat model" was obtained in each case. Both threat models consist of the possible ways in which the voter could be coerced, and, as might be expected, vary for each scheme. These requirements were then informally expressed in terms of opacity.

There were four cases to consider in Prêt à Voter, and three in the FOO scheme, each taking into account different capabilities of the coercer. In Prêt à Voter, the requirement for coercion-resistance, informally expressed as opacity, is that the association between the onion and corresponding candidate list should be opaque to the coercer. In the FOO scheme the requirement is that the random factor used by the voter to commit her vote should be opaque to the coercer. As shown above, this only holds in the case of a "passive" coercer, and is not otherwise preserved by the protocol. The two opacity statements highlight the difference between the two schemes: in FOO, the voter participates in a cryptographic exchange with the system, whereas this is not the case in Prêt à Voter. To derive the opacity statements in each case, the predicate, $\phi$, has been defined according to the property that should be opaque to the coercer.

A formal definition of receipt-freeness cast as a variation of the opacity property was derived from the intuition gained by the analysis [138]. This is the first step towards a definition of coercion-resistance as opacity, for which there are additional considerations to take into account, as will be discussed shortly.

The basic intuition of the following definition of receipt-freeness cast as a variation of opacity [138] is that even if the voter is prepared to reveal her secrets to the coercer, she should not be able to prove which way she voted. Let $\lambda_{v,k}$ denote a run in which the voter casts a vote $v$, using a key, $k$. Let $S$ be the system, $C$ the set of available candidates and $K$ the set of available keys.

$$\forall \lambda_{v,k} \in \tau(S),\ \forall v' \in C,\ k \in K,\ \exists k' \in K, \text{ such that } \lambda_{v',k'} \in \tau(S) \text{ and}$$

$$obs_{k'}(\lambda_{v,k}) = obs_{k'}(\lambda'_{v',k'}),$$

where $obs_k$ denotes the observation function that the coercer can apply to his view of the election protocol, "informed" by his "knowledge" of $k$. By the definition above, he is unable to tell whether the voter submitted $v$ using a key $k$, or the coercer's vote choice, $v'$ using a key, $k'$, which has been "revealed" to the coercer.

This is a generic definition of receipt-freeness, which can be specialised according to a particular voting scheme. A model is then needed for the scheme and $obs_k$ must be defined. Note that this definition is not a direct casting of receipt-freeness as opacity, as the predicate, $\phi$, has not been defined. It could be regarded as an "unwinding" of the stricter form, and would be easier to use in practice.

For example, from the analysis of Prêt à Voter, in terms of opacity, $\phi$ was defined as being true of a trace if it contains if it contains the association between the onion and corresponding candidate list, which should be opaque to the coercer. Similarly, in the FOO scheme, $\phi$ was defined as being true of a trace if it contains the random factor used to commit a vote. Basing the model on either scheme, $\phi$ and could be denoted by $k$ in the definition above.

This definition captures receipt-freeness rather than coercion-resistance as the attacker model is "passive", that is, the coercer can only observe instances of the protocol. For a definition of coercion-resistance, it would be necessary to model an "active" coercer, that is, one able to influence the target voter's interaction with the system. This could, for example, be done using a suitably defined *obs* function, and is the subject of current research.

## 7.6 Related Primitives

*Designated verifier proofs* [88] and *deniable encryption* [30], have been both mentioned in the literature on voting systems. In this section, I discuss the possible relationships between each of these primitives and the opacity property.

### 7.6.1 Designated Verifier Proofs

*Designated verifier proofs* (DVPs) were proposed by Jakobssen et al in [88]. Intuitively, a designated verifier proof one in which Alice proves to Bob, and only to Bob, that the statement $\theta$ is true. Let $\Phi_B$ be the statement "I know Bob's secret key". Alice proves the statement $\theta \vee \Phi_B$ to Bob, who will be convinced that $\theta$ is true (or that his secret key has been compromised). However, Bob cannot use this proof to convince Cindy that $\theta$ is true, even if he shares his secret key with her.

DVPs have been used to strengthen coercion-resistance in several voting protocols, for example, [136], [91], [82].) In a typical example, the authority sends the voter her voting keys using a DVP so that only the voter knows that it is valid, hence, evading the potential coercer. As another example, in a remote implementation of Prêt à Voter [38], the server sends a ballot form, $b$, to the voter with a designated verifier proof. Only the voter can verify that $b$ is really her form. This eliminates the need for either the voting booth or shredder, which, in Prêt à Voter "classic" [36], are methods to erase any association between the voter and her vote. Recall that the candidate list is printed on the left-hand side of a ballot form, the onion on the right. In the booth, the voter marks her selection on the right-hand side against her chosen candidate, and separates the left- and right-hand strips. The left-hand strip is then destroyed, for example, by feeding it into a shredder, while the right-hand strip is retained as a receipt. Without the corresponding left-hand strip, the vote cannot be verified by anyone other than the voter, as she alone knows the candidate ordering associated with the onion value on her receipt.

An alternative way to define a DVP, also given in [88], indicates a relationship with a variation of opacity.

"Let $(P_A, P_B)$ be a protocol for Alice to prove the truth of the statement $\theta$ to Bob. Bob is a designated verifier if the following is true: for any protocol $(P_A, P'_B, P_C)$ involving Alice, Bob and Cindy, in which Bob proves the truth of $\vartheta$ to Cindy, there is another protocol $(P''_B, P_C)$ such that Bob can perform the calculations of $P''_B$ and Cindy cannot distinguish transcripts of $(P_A, P'_B, P_C)$ from those of $(P''_B, P_C)$".

A possible re-statement in terms of opacity is given below.

Suppose a run of the system in which Alice proves a statement $s$ to Bob, and with the predicate $\phi$ defined as being true of a trace if it contains $s$. Bob is the designated verifier if, for any subsequent runs of the system involving Alice, Bob and Cindy, in which Bob tries to prove $s$ to Cindy, if $\phi$ is true in one run, then there is another run in which $\phi$ is false, and Cindy cannot distinguish between them.

In other words, $\phi$ is opaque to Cindy, and only Bob can verify $s$, as intended by Alice.

## 7.6.2 Deniable Encryption

Proposed by Canetti et al in [30], *deniable encryption* is a scheme primarily designed to safeguard the exchange of encrypted messages. The essential idea is to prevent an attacker demanding key information from either the sender or receiver, or both. More precisely, the sender (or both sender and receiver) can deceive the attacker by providing a "fake" decryption of the original message. More precisely,

"Suppose Alice sends a ciphertext $c = E(m_1, r)$, where $m_1$ is some message, $E$ is a public encryption algorithm, and $r$ is Alice's local random input. Can Alice come up with a fake random input $r$' that will make $c$ 'look like' an encryption of a different message $m_2$? We call encryption schemes that have this property deniable." [30].

In other words, if the attacker demands the secret key, one can be provided such that the same ciphertext reveals a different message, hence preserving the secrecy of the original plaintext. A good analogy of this is the one-time pad, which has the property that for a given ciphertext, a different plaintext from the original can always be found.

Casting deniable encryption as opacity is straightforward. Informally, $\phi$ could be defined as being true of a trace if it contains the value of the (encrypted) plaintext, which should be opaque to the observer.

Note that for a fuller treatment of both DVPs and deniable encryption, one would need to define a suitable *obs* function. This is, however, outside the present scope.

At first glance, it might seem that deniable encryption could provide a viable solution to coercion in voting schemes. As described previously, Delaune et al [46] propose a scheme in which the voter can use either a "fake" or a valid key to encrypt her vote, and the coercer cannot distinguish between the two encryptions. However, it is observed, although not fully elucidated in [91] and [82] that deniable encryption schemes do not provide coercion-resistance. The reason, later explained in personal communication with Canetti, is that not all voters will be honest. Depending on the protocol, it may be possible for the voter to render the scheme "undeniable", for example, by choosing the randomness used for encryptions so that the result is publicly verifiable.

One way to get around this is for the authority to distribute key material to voters via untappable channels, so that the keys cannot not be intercepted by the coercer. Alternatively, sensitive voter interaction could take place in a booth. This perhaps highlights the necessity for designing voting protocols with care, particularly those in which voters participate in a cryptographic exchange with the system. This has been amply demonstrated in previous chapters with the Chaum, Neff and FOO schemes.

Despite its drawbacks, deniable encryption could still be useful in certain situations. Interestingly, deniable encryption appears to be the logic behind some coercion-resistant voting systems, but is reinforced with other schemes such as DVPs. An example is the scheme proposed by Delaune et al [46], in which the voter is able to cast a vote encrypted with either a "fake" or true key, which are initially issued by the authority using a DVP.

## 7.7 Anonymity

Coercion-resistance has been related to a stronger form of secrecy [46] in that not only should the value of the voter's vote be secret, but, as the voter may volunteer information to the coercer, the act of casting the vote should be secret as well. If resistance to forced abstention is also required, then the set of information to be kept secret should include the voter's identity.

Another possible way of expressing coercion-resistance is by using anonymity. Schneider et al [176] have defined strong anonymity in the process algebra CSP [83]. By their definition, a process is strongly anonymous on a set, $A$, if it makes all events from $A$ available whenever any such event is possible. $A$ could be regarded as the set of information that is to be kept anonymous, and hence, varied depending on the requirements of the system.

Following the intuition of [46] above, coercion-freeness could be expressed as a requirement for the value of a vote and the voter's identity to be anonymous. If forced abstention is required, then the act of casting a vote could be added to the set of information to be anonymous. Note that this definition does not capture resistance to simulation or randomisation attacks. Translated into opacity, the predicate $\phi$, defined as being true of a trace if it contains the value of a vote, identity and the act of casting a vote associated with a particular voter, should be opaque with respect to a suitably defined *obs* function. Once again, this demonstrates the flexibility of opacity as the predicate, $\phi$, and *obs* can be defined according to the particular requirements. Defining different forms of anonymity in terms of opacity is, however, the subject of future research.

As shown above, various notions of coercion-resistance can be related to anonymity over system properties such as voter identity and the value of a vote.

## 7.8 Discussion and Summary

Coercion-resistance is an essential requirement for voting systems as it ensures several other desirable properties, such as secrecy, legitimacy and eligibility. It is therefore of no surprise that there is currently much interest in this topic in the field, and several coercion-resistant voting systems have been proposed. Some of this interesting and valuable work has been discussed in this chapter, and extended with an analysis of coercion-resistance in two markedly different voting systems, Prêt à Voter and the FOO scheme.

The results of this analysis tallies with those obtained in the analyses carried out in Chapters 5 and 6. Prêt à Voter has a high resistance to coercion, mainly due to the fact that random inputs are not required from the voter or voting device in receipt preparation, and that voting is supervised. In the Chaum and Neff schemes, as described in [92], these random choices open up the possibility of coercion via semantic or random subliminal channels. However, in Prêt à Voter, the ballot forms are prepared ahead of time, and an unmarked form which falls into the hands of a third party could be used to initiate a chain-voting attack. This could be regarded as a form of coercion, and counter-measures, such as concealing the onion until the vote is actually cast, have been discussed in [163]. In the FOO scheme, voters participate in ballot creation, and hold the keys used to encrypt and verify their votes. Coercion is a problem, as the keys could fall into the wrong hands.

This is believed to be the first analysis of coercion-resistance in the FOO scheme, and the parallel analysis of Prêt à Voter extends earlier work presented in Chapters 5 and 6. In addition, coercion-resistance has been examined from a new angle, that is, in terms of opacity, which indicates a close relationship between the two properties. A formal definition has been proposed for receipt-freeness cast as opacity [138]. Possible applications of this definition have been discussed, using Prêt à Voter and the FOO scheme as examples, along with future extension to capture the stronger notion of coercion-resistance.

Voting systems vary widely in structure and operation, and likewise their requirements for coercion-resistance. Due to its generality, opacity has the flexibility to accommodate these differences. For example, coercion-resistance requirements in both Prêt à Voter and the FOO scheme can be expressed, in each case, by altering the predicate, $\phi$, over which opacity is defined. This was carried out informally, but a full instantiations are planned along with a formal definition of coercion-resistance as opacity in future work.

Designated verifier proofs, deniable encryption have both been discussed in terms of variations of opacity. There appears to be a relationship between coercion-resistance and anonymity. Future work on defining different forms of anonymity in the opacity framework, may offer an alternative to the casting coercion-resistance as a variation of general opacity.

As well as contributing to research on coercion-resistance, the work presented in this chapter is further demonstration of the opacity property as a highly useful and flexible tool for security analysis. It may also exemplify the composition of a systems- and information-flow analysis, and could be a useful part of future work on a more systematic approach to security analysis than that performed, for example, in [92] and [163].

# Chapter 8

# Conclusion

Opacity has been investigated in relation to other existing information flow properties, and its extreme flexibility demonstrated in various practical applications. In this chapter, I give a summation of the work achieved and finally, a view of the thesis in its entirety.

Possible links between variations of opacity and several existing security properties were discussed in Chapter 2. The Purge Closure Property(PCP) was defined and used to prove that non-interference implies opacity, but that the converse is not true. The PCP states that for all traces of the system, $S$, the purge of a trace is a valid trace of $S$, where "purge" means that all High level events have been removed. By linking the PCP to the purge function formulated by McLean [118], opacity was shown to imply non-inference, and vice versa [139]. Non-deducibility and (a special case of) non-leakage were both cast as opacity [139]. Since non-interference cannot be directly cast as opacity, the same is true for non-influence, which is defined as non-interference plus non-leakage [185].

Strictly speaking in the relationships described above, I refer to specialisations of opacity. Recall that opacity is parametric in that the predicate, $\phi$ and the observation function can be defined to reflect the security concerns of a system. For example, in the study of information flow properties, $\phi$ was, in many cases, defined as being true of a trace if it contains High-level events. However, for non-deducibility, it was defined as being true of a trace if it does not contain High-level events. In the case of non-leakage, opacity was defined over the set of initial states of the system. Note that non-leakage, like its related property, non-influence, is designed to capture intransitive policies, whereas so far, opacity has only been defined for transitive information flow. However, it has been shown that transitive is a special case of intransitive flow [185]. Hence, in mapping this property to opacity, I refer also to a variation of non-leakage.

Based mainly on Prêt à Voter, a case study consisting of three different forms of security analysis, has been carried out. The first, described in Chapter 5, analysed the information flow in Prêt à Voter, and, to provide a comparison, the FOO scheme. It was found that many of the security requirements of both schemes could be expressed as variations of opacity. The exceptions were cases where safety [172] and liveness [12] properties were found to be more appropriate than information flow properties. As discussed earlier, the difference between safety and information flow properties is that safety properties can be expressed as predicates on system traces, whereas information flow properties describe a "space of behaviours" [160]. Like safety properties, liveness is also defined on system traces [12]. However, while safety properties have been shown to be "enforceable", this is not true of either information flow or liveness properties [172], [161]. A further treatment of safety and liveness properties is beyond the present scope, but would be an interesting future extension to the analysis.

In [92] Karlof et al analyse the Chaum and Neff voting schemes, and identify various weaknesses due to the interactions between the various components of each system, such as the hardware, software and human players. Ryan et al extend this work with a systems-based analysis of Prêt à Voter in [163], which is detailed in Chapter 6. In this analysis, information flow was examined at a higher level of abstraction than in the previous chapter. Prêt à Voter was found to be robust against many of the vulnerabilities mentioned in [92], although chain-voting is a potential problem. As with the other vulnerabilities, counter-measures are suggested, for instance, concealing the onion value on the ballot form until the a vote is actually cast. The analysis also highlighted areas for future enhancement of the scheme, such as distributed ballot creation to avoid single authority knowledge of ballot form information. A possible scheme is described in [165].

Although extremely useful in locating potential vulnerabilities, the analysis of [92] and [163] could not be regarded as systematic, and perhaps does not provide exhaustive results. A more thorough approach might be achieved by combining it with an analysis of information flow, such as the one carried out Chapter 5. As an illustration, chain-voting is analysed in both forms, the first at design-level the second, in the way the "environment", namely the coercer, could interfere with the protocol. This would be an interesting subject for future work.

Coercion-resistance is an essential property of voting systems, without which other requirements, such as ballot secrecy and legitimacy may fail. Not surprisingly, it has received much interest in the field and formalisations of the property have been proposed [91], [46], as well as several voting schemes that strive for coercion-resistance [38], [82], [21]. It usually describes the inability of a voter to prove her vote, and is sometimes referred to as receipt-freeness. However, the latter term can be misleading. For instance, in the FOO scheme, the voter does not receive a physical receipt, and yet is vulnerable to coercion. On the other hand, Prêt à Voter is fairly resistant to coercion, even though the voter does receive a receipt. Juels et al define coercion-resistance as a stronger property than receipt-freeness, as it includes resistance to forced abstention, simulation and randomisation attacks as well as the inability to prove a vote [91]. Delaune et al also define coercion-resistance as a stronger property in that it implies receipt-freeness [47].

In Chapter 7, Prêt à Voter and the FOO scheme were analysed for coercion-resistance [138]. It was shown that, in both schemes, the requirements for coercion-resistance can be informally expressed in terms of a suitably defined form of opacity. This leads to a formalisation of receipt-freeness cast as opacity [138]. This definition could be reinforced to capture coercion-resistance by modelling an "active" attacker, perhaps with an appropriate *obs* function, and is the subject of future research. In addition to having the ability to express this property in a succinct way, the opacity property appears to be flexible enough to capture the requirements of two markedly different schemes. It is interesting to note that in this analysis, information flow is examined at both protocol- and component-level, and would be a useful example for future work on a more systematic approach to security analysis than that performed in [92] and [163].

Together, the results obtained from the three forms of analysis presented in this thesis make a strong case for opacity as a useful framework for security analysis of systems. As mentioned previously, the systems-based analysis has indicated a "taxonomy of vulnerabilities", which may be starting-point for a more thorough and methodical approach.

I now review other areas of possible future work. Opacity is defined in over an observational equivalence, which, as discussed previously, is closer to trace equivalence in the semantics of process algebras, such as CSP. In process algebraic terms, two processes are observationally equivalent if they have the same behaviours, taking into account internal events and the way non-determinism is resolved [83]. Observational equivalence, in this sense, is stronger than trace equivalence, which only accounts for the visible events. Stronger security properties can be defined around an attacker able to perform, or choose not to perform actions, as well as to observe them. The ability to distinguish different non-deterministic behaviours is the intuition behind the "testing" and "refusals" semantics of CSP [160], [164]).

Recall that, in opacity the *obs* function can be either label-oriented, state-oriented or both. It is feasible that a state- and label-oriented *obs* could be used for stronger equivalence models. Essentially, this would model a more powerful observer who may be able to distinguish between events, as well as result of events. Stated in terms of opacity, it may thus be possible to obtain more robust definitions of security than by using a label-based *obs* alone. This is a possible line of further work.

It is possible that a dynamic state- and label-oriented *obs* function could also model intransitive flow, or, more precisely, the visibility aspects of the system depending on the state and/or labels leading to, possibly even from, that state. The ability to capture intransitive flow would be a useful addition to the opacity framework. Presently, "dynamic" information flow could be modelled by defining the predicate, $\phi$, as necessary for security of the system.

Another area for possible further research is the compositionality aspects of opacity. Compositionality is essential for verifying that systems composed from secure components are secure. It is well known that non-deterministic information flow properties are generally not compositional. Unlike safety [172] and liveness [12] which are trace properties, information flow properties are defined over *sets of traces* [118], [160]. Hence they tend to defy the Abadi-Lamport Composition Principle [9], which only holds for safety and liveness properties. There are, however, exceptions to this rule, for example, process algebraic definitions of non-interference in [55], [160], [175]. It is worth noting, however, that in most cases, proofs for compositionality only hold for a certain set of specified constructs, usually given at the outset. It would be useful and interesting to find out whether or not opacity is compositional, and if so, to which extent. A feature of software engineering is modularity, for example, the ability to build systems from COTS products, and with the growing demand for secure software, compositional security properties would be highly desirable.

As mentioned earlier, the work of Halpern et al [77], [78] shows that certain security properties are weakened when translated from a synchronous to asynchronous model. It would be interesting to determine the strengths and weaknesses of a synchronous form of opacity in relation to the current asynchronous version. It is possible that synchronous opacity could be modelled by using a state- and/or label-based *obs* function, and including time in state information.

In [96], Lakhnech et al define probabilistic opacity in a cryptographic setting. Although they claim that it readily maps to "classical" opacity, this has yet to be verified in the present framework.

This thesis provides evidence that the opacity property has great application in two important areas of security work. Firstly, in formalising the security requirements and secondly, in analysing the security properties of practical systems. In addition, the extreme flexibility of opacity has been demonstrated in its ability to capture a wide range of different security requirements. Based on this work, I am confident that further development of the present framework would certainly be worthwhile.

In total, there is strong indication that opacity is an extremely valuable tool for the specification and analysis of high assurance systems such as voting schemes.

# Bibliography

[1] Defence Science and Technology Laboratory. http://www.dstl.gov.uk/index.php.

[2] Engineering and Physical Sciences Research Council. http://www.epsrc.ac.uk/default.htm.

[3] The Interdisciplinary Research Collaboration in Dependability. http://www.dirc.org.uk/.

[4] LaTeX - a document preparation system. http://www.latex-project.org/.

[5] VoteHere. http://www.votehere.net/default.php.

[6] The trouble with technology. The Economist, Sept 16 2004. http://economist.com/World/na/displayStory.cfm?story_id=3195821.

[7] Voting systems, 2006. http://en.wikipedia.org/wiki/Category:Voting_systems.

[8] M. Abadi and C. Fournet. Mobile values, new names, and secure communications. In *Symposium on Principles of Programming Languages*, page 104.

[9] M. Abadi and L. Lamport. Composing specifications. Technical Report 66, Digital Equipment Corporation Systems Research Center, Palo Alto, CA, 1990.

[10] S. Abramsky and A. Jung. Domain theory. In *Handbook of Logic in Computer Science*, volume 3. Oxford University Press, 1994.

[11] P. G. Allen. A comparison of non-interference and non-deducibility using CSP. In *IEEE Computer Security Foundations Workshop*, 1991.

[12] B. Alpern and F. Schneider. Liveness. Technical Report TR-85-66, Cornell University. 1996.

[13] The Anonymizer. http://anonymizer.com.

[14] W. R. Ashby. *An Introduction to Cybernetics.* Chapman & Hall, London, 1956.

[15] M. Backes and B. Pfitzmann. Intransitive non-interference for cryptographic purposes. In *Symposium on Security and Privacy.* IEEE, 1991.

[16] M. Backes and B. Pfitzmann. Computational probabilistic non-interference. In *European Symposium on Research in Computer Security*. IEEE, 2002.

[17] J. Bannet, W. Price, A. Rudys, J. Singer, and D. Wallach. Hack-a-vote: Security issues with electronic voting systems. *IEEE Security and Privacy*, 2(1), Jan/Feb 2004.

[18] O. Baudron, P.-A. Fouque, D. Pontecheval, G. Poupard, and J. Stern. Practical multi-candidate election system. In *Symposium on Principles of Distributed Computing*, pages 274–283. ACM, 2001.

[19] D. Bell and L. LaPadula. Secure computer system: a mathematical model. Technical Report TR-2547, MITRE, 1973. *Reprinted in Journal of Computer Security, Vol. 4, No. 2-3, 1996.*

[20] D. Bell and L. LaPadula. Secure computer system: Unified exposition and multics interpretation. Technical Report ESD-TR-2997, MITRE, 1976.

[21] J. Beneloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Symposium on Theory of Computing*, pages 544 – 553. ACM, 1994.

[22] M. Bhargava and C. Palamidessi. Probabilistic anonymity. Technical report, INRIA Futurs and LIX, 2005.

[23] K. J. Biba. Integrity considerations for secure computer systems. Technical Report ESD-TR-76-372, USAF Electronics Division, 1977.

[24] A. Bossi, C. Piazza, and S. Rossi. Modelling downgrading in information flow security. In *IEEE Computer Security Foundations Workshop*, 2004.

[25] J. Bryans, M. Koutny, Laurent Mazaré, and P.Y.A. Ryan. Opacity generalised to transition systems. Technical Report CS-TR-868, University of Newcastle upon Tyne, 2004.

[26] J. Bryans, M. Koutny, and P.Y.A. Ryan. Modelling dynamic opacity using Petri nets with silent actions. In *Formal Aspects of Security and Trust*, 2004.

[27] J. Bryans, M. Koutny, and P.Y.A. Ryan. Modelling opacity using Petri nets. In *Workshop on Security Issues with Petri Nets and other Computational Models*, 2004.

[28] J. Bryans and P.Y.A. Ryan. A dependability analysis of the Chaum voting scheme. Technical Report CS-TR-809, University of Newcastle upon Tyne, 2003.

[29] N. Busi and R. Gorrieri. Structural non-interference with Petri nets. In *Workshop on Issues in the Theory of Security*, 2004.

[30] R. Canetti, C. Dwork, and M. Naor. Deniable encryption. In *CRYPTO '97*, number 1294 in Lecture Notes in Computer Science, pages 90–104. Springer-Verlag, 1999.

[31] K. Chatzikolakis and C. Palamidessi. Probable innocence revisited. In *Formal Aspects in Security and Trust*, 2005.

[32] D. Chaum. Untraceable mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[33] D. Chaum. Security without identification : Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[34] D. Chaum. The Dining Cryptographer's Problem: Unconditional sender and recipient anonymity. *Journal of Cryptology*, 1, 1988.

[35] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, January-February 2004.

[36] D. Chaum, P.Y.A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *European Symposium on Research in Computer Security*, number 3679 in Lecture Notes in Computer Science. Springer-Verlag, 2005.

[37] S. Chong and A. Myers. Language-based information erasure. In *IEEE Computer Security Foundations Workshop*. IEEE, 2005.

[38] M. Clarkson and A. Myers. Coercion-resistant remote voting using decryption mixes. In *Workshop on Frontiers of Electronic Elections*, 2005.

[39] E. Cohen. Information transmission in computational systems. In *Symposium on Operating Systems Principles*. ACM, 1977.

[40] F. Cuppens. A logical anaysis of authorised and prohibited information flows. In *Symposium on Security and Privacy*. IEEE, 1993.

[41] F. Cuppens. A logical formalization of secrecy. In *IEEE Computer Security Foundations Workshop*. IEEE, 1993.

[42] F. Cuppens and P. Bieber. A definition of secure dependencies using the logic of security. In *IEEE Computer Security Foundations Workshop*. IEEE, 1991.

[43] F. Cuppens and P. Bieber. A logical view of secure dependencies. *Journal of Computer Security*, 1(1), 1992.

[44] F. Cuppens and P. Bieber. Secure dependencies with dynamic level assignments. In *IEEE Computer Security Foundations Workshop*. IEEE, 1992.

[45] F. Cuppens and G. Trouessin. Information flow controls vs inference controls: an integrated approach. In *European Symposium on Research in Computer Security*, number 875 in Lecture Notes in Computer Science. Springer-Verlag, 1994.

[46] S. Delaune, S. Kremer, and M. Ryan. Receipt-freeness: formal definition and fault attacks. Workshop on Frontiers of Electronic Elections, to appear, 2005.

[47] S. Delaune, S. Kremer, and M. Ryan. Coercion-resistance and receipt-freeness in electronic voting. Technical Report LSV-06-08, Laboratoire Spécification et Vérification, 2006.

[48] Y. Desmedt and K. Kurosawa. How to break a practical mix and design a new one. In *EUROCRYPT*, 2000.

[49] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Symposium on Theory of Computing*, pages 542–552. ACM, 1991.

[50] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[51] Formal Systems Europe. http://www.fsel.com/software.html.

[52] R. Feiertag. A technique for proving specifications are multilevel secure. Technical Report CSL-109, SRI International, 1996.

[53] R. Focardi. Comapring two information flow security models. In *Symposium on Security and Privacy*. IEEE, 1996.

[54] R. Focardi and R. Gorrieri. An information flow property for CCS. In *North American Process Algebra Workshop*, 1993.

[55] R. Focardi and R. Gorrieri. Classification of security properties (part 1: Information flow). In *Foundations of Security Analysis and Design*, volume 3, 2000.

[56] R. Focardi, R. Gorrieri, and Fabio Martinelli. Non-interference for the analysis of cryptographic protocols. In *International Colloquium on Automata, Languages, and Programming*, number 1858 in Lecture Notes in Computer Science. Springer-Verlag, 2000.

[57] S. N. Foley. A universal theory of information flow. In *Symposium on Security and Privacy*. IEEE, 1987.

[58] R. Forster. *Non-interference Properties for Nondeterministic Processes*. PhD thesis, Oxford University Computing Laboratory, 1987.

[59] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, pages 244–251. ACM, 1992.

[60] T. El Gamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469472, 1895.

[61] F. D. Garcia, W. Pieters I. Hasuo, and P. van Rossum. Provable anonymity. In *Workshop on Formal Methods in Security Engineering*. ACM, 2005.

[62] P. Gardiner. Power simulation and its relation to traces and failures refinement. *Electronic Notes in Theoretical Computer Science*, 32, 2000.

[63] E. Gerck. Instant lottery cards too, re: reading pins in 'secure' mailers without opening them, 2005. http://www.mail-archive.com/cryptography

[64] J. Glasgow, G. McEwen, and P. Panangaden. A logic for reasoning about security. In *IEEE Computer Security Foundations Workshop*. IEEE, 1990.

[65] M. Gogolewski, M. Klonowski, P. Kubiak, M. Kutylowski, A. Lauks, and F. Zagorski. Kleptographic attacks on e-election schemes with receipts. In *International Conference on Emerging Trends in Information and Communication Security*, Lecture Notes in Computer Science. Springer-Verlag, 2006.

[66] J. Goguen and J. Meseguer. Security policies and security models. In *Symposium on Security and Privacy*. IEEE, 1982.

[67] J. Goguen and J. Meseguer. Unwinding and inference control. In *Symposium on Security and Privacy*. IEEE, 1984.

[68] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity. *Journal of the ACM*, 38(3):690–728, 1991.

[69] P. Golle and M. Jakobsson. Reusable anonymous return channels. In *Workshop on Privacy in the Electronic Society*, 2003.

[70] J. Graham-Cumming. *The Formal Development of Secure Systems*. PhD thesis, Oxford University Computing Laboratory, 1992.

[71] J. Graham-Cumming and J. W. Sanders. On the refinement of non-interference. In *IEEE Computer Security Foundations Workshop*, 91.

[72] J. W. Gray. Toward a mathematical foundation for information flow security. In *Symposium on Security and Privacy*. IEEE, 1991.

[73] J. W. Gray and P. F. Syverson. A logical approach to multilevel security of probabilistic systems. In *Symposium on Security and Privacy*. IEEE, 1992.

[74] C. Gülcü and G. Tsudik. Mixing e-mail with Babel. In *Network and Distributed Security Symposium*, 1996.

[75] J. Guttman and M. Nadel. What needs securing? In *IEEE Computer Security Foundations Workshop*, pages 34–57. IEEE, 1988.

[76] T. Haigh and W. Young. Extending the non-interference version of MLS for SAT. In *Symposium on Security and Privacy*. IEEE, 1986.

[77] J. Halpern and K. O'Neill. Secrecy in multiagent systems. *Submitted for publication March 2005. Available from http://www.cs.cornell.edu/people/oneill/.*

[78] J. Halpern and K. O'Neill. Secrecy in multiagent systems. In *IEEE Computer Security Foundations Workshop*, 2002.

[79] J. Halpern and K. O'Neill. Anonymity and information hiding in multiagent systems. In *IEEE Computer Security Foundations Workshop*, 2003.

[80] M. Hennessey. The security picalculus and non-interference. *Journal of Logic and Algebraic Programming*, 63(1):3–34, 2004.

[81] J. Hintikka. *Knowledge and Belief.* Cornell University Press, 1963.

[82] Martin Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptology*, number 1807 in Lecture Notes in Computer Science, pages 539–556. Springer-Verlag, 2000.

[83] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[84] D. Hughes and V. Schmatikov. Information hiding, anonymity and privacy: A modular approach. *Journal of Computer Security*, 12(1):3–36, 2002.

[85] J. Jacob. On the derivation of secure components. In *Symposium on Security and Privacy.* IEEE, 1989.

[86] J. Jacob. Specifying security properties. In C. A. R. Hoare, editor, *Developments in Concurrency and Communication.* Addison-Wesley, 1990.

[87] M. Jakobsson, A. Juels, and Ronald Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX Security Symposium*, pages 339–353, 2002.

[88] M. Jakobsson, K. Sako, and R. Impagiazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, number 1070 in Lecture Notes in Computer Science, pages 143–154. Springer-Verlag, 1996.

[89] D. Johnson and F. Thayer. Security and the composition of machines. In *IEEE Computer Security Foundations Workshop*, pages 72–89. IEEE, 1988.

[90] D. W. Jones. A brief illustrated history of voting, 2003. http://www.cs.uiowa.edo/ jones/voting/pictures.

[91] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. To appear, 2002.

[92] C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: A systems perspective. In *USENIX Security Symposium*, 2005.

[93] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach. Analysis of an electronic voting system. In *Symposium on Security and Privacy*. IEEE, 2004.

[94] S. Kremer and M. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *European Symposium on Programming*, number 3444 in Lecture Notes in Computer Science, pages 186–200. Springer-Verlag, 2005.

[95] S. Kripke. Semantic analysis of modal logic. I: Normal propositional calculi. *Zeitschrift Mathematische Logik und Grundlagen der Mathematik*, 9(67-96), 1963.

[96] Y. Lakhnech and L. Mazaré. Probabilistic opacity for a passive adversary and its application to chaum's voting scheme. Technical Report TR-2005-4, Verimag, 2005.

[97] L. Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, 3(2):123–145, 1977.

[98] B. Lampson. Protection. In *Princeton Symposium on Information Sciences and Systems*, 1971.

[99] B. Lampson. A note on the confinement problem. *ACM Operating Systems Review*, 16(10):613–615, 1973.

[100] T. W. Lauer. The risk of e-voting. *Electronic Journal of e-Government*, 2(3), Dec 2004.

[101] G. Lowe. Defining information flow. Technical report, Leicester University, 1993.

[102] H. Mantel. Possibilistic definitions of security - an assembly kit. In *IEEE Computer Security Foundations Workshop*. IEEE, 2000.

[103] H. Mantel. Unwinding possibilistic security properties. In *European Symposium on Research in Computer Security*, 2000.

[104] H. Mantel. Quantifying information flow. In *Symposium on Security and Privacy*. IEEE, 2002.

[105] H. Mantel. *A Uniform Framework for the Formal Specification and Verification of Information Flow Security*. PhD thesis, University of Saarlande, 2003.

[106] H. Mantel and D. Sands. Controlled declassification based on intransitive non-interference. In *Asian Symposium on Programming Language and Systems*, number 3302 in Lecture Notes in Computer Science, pages 129–145. Springer-Verlag, 2004.

[107] L.F. Marshall and C. Molina-Jiminèz. Anonymity with identity escrow. In *International Workshop on Formal Aspects in Security and Trust*, 2003.

[108] L. Mazaré. Using unification for security properties. In *Workshop on Issues in the Theory of Security*, 2004.

[109] D. Mazières and M. F. Kaashoek. The design, implementation and operation of an email pseudonym server. In *Computer and Communications Security*. ACM, 1998.

[110] D. McCullough. Specifications for multi-level security and a hook-up property. In *Symposium on Security and Privacy*. IEEE, 1987.

[111] D. McCullough. Covert channnels and degrees of insecurity. In *IEEE Computer Security Foundations Workshop*. IEEE, 1988.

[112] D. McCullough. Ulysses: A computer security modelling environment. In *National Computer Security Conference*, 1988.

[113] J. McHugh. Covert channel analysis. In *Handbook for the Computer Security Certification of Trusted Systems*. Naval Research Laboratoty, Washington, D.C., 1995.

[114] J. McLean. A comment on the basic security requirements of bell and lapadula. *Information Processing Letters*, 20(2), 1985.

[115] J. McLean. Security models. In *Encyclopedia of Software Engineering*. Wiley, 1985.

[116] J. McLean. Reasoning about security models. In *Symposium on Security and Privacy*. IEEE, 1987.

[117] J. McLean. Security models and information flow. In *Symposium on Security and Privacy*. IEEE, 1990.

[118] J. McLean. A general theory of trace sets closed under selective interleaving functions. In *Symposium on Security and Privacy*. IEEE, 1994.

[119] R. Mercuri. A better ballot box? *IEEE Spectrum Online*, October 2002.

[120] J-J. Ch. Meyer and R. J. Wieringa. Deontic logic: a concise overview. In *Deontic Logic in Computer Science*. Wiley, 1993.

[121] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[122] R. Milner. *Communicating and Mobile Systems: the $\pi$-Calculus*. Cambridge University Press, 1999.

[123] C. Molina-Jiminèz and L.F. Marshall. True anonymity without mixes. Technical Report CS-TR: 727, University of Newcastle upon Tyne, 2005.

[124] I. Moskowitz. Quotient states and probabilistic channels. In *IEEE Computer Security Foundations Workshop*. IEEE, 1990.

[125] I. Moskowitz and O. Costich. A classical automata approach to noninterference type problems. In *IEEE Computer Security Foundations Workshop*. IEEE, 1992.

[126] M. Mukhamedov and M.D. Ryan. On anonymity with identity escrow. In *International Workshop on Formal Aspects in Security and Trust*, 2005.

[127] J. Mullins. Nondeterministic admissible interference. *Journal of Universal Computer Science*, 6(11), 2000.

[128] A. Myers and B. Liskov. Protecting privacy using the decentralised label model. *ACM Transactions on Software Engineering and Methodology*, 9(4), 2000.

[129] M. Naor. Bit commitment using pseudo-randomness. In *Advances in Cryptography*, number 435 in Lecture Notes in Computer Science, pages 128–136. Springer-Verlag, 1989.

[130] M. Naor and A. Shamir. Visual cryptography. In *Advances in Cryptology*, number 950 in Lecture Notes in Computer Science, pages 1–12. Springer-Verlag, 1994.

[131] A. Neff. A verifiable secret shuffle and its application to e-voting. In *Conference on Computer and Communications Security*, pages 116–125. ACM, 2001.

[132] A. Neff. Practical high certainty intent verification for encrypted votes, 2004. http://www.votehere.net/documentation/vhti.

[133] A. Neff. Verifiable mixing (shuffling) of el-gamal pairs, 2004. http://www.votehere.net/documentation/vhti.

[134] C. O'Halloran. A calculus of information flow properties. In *European Symposium on Research in Computer Security*, 1990.

[135] T. Okamoto. An electronic voting scheme. In *WIFIP World Conference on IT Tools*, pages 21–30, 1996.

[136] T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Workshop on Security Protocols*, number 1361 in Lecture Notes in Computer Science, pages 25–35. Springer-Verlag, 1997.

[137] C. Palamidessi. Probabilistic and nondeterministic aspects of anonymity. In *Mathematical Foundations of Programming Semantics*, 2005.

[138] T. Peacock and P.Y.A. Ryan. Coercion-resistance as opacity in voting systems. Technical Report CS-TR-959, University of Newcastle upon Tyne, 2006.

[139] T. Peacock and P.Y.A. Ryan. Opacity - further insights on an information flow property. Technical Report CS-TR-958, University of Newcastle upon Tyne, 2006.

[140] A. Pfitzmann, M. Kohntopp, and A. Shostack. Anonymity, (unlinkability,) unobservability, pseudonymity, and identity management - a (consolidated) proposal for terminology. http://www.dud.inf.tu-dresden.de/Literatur_V1.shtml, 2005.

[141] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *GI/ITG Conference on Communication in Distributed Systems*, 1991.

[142] B. Pfitzmann and A. Pfitzmann. How to break the direct RSA-implementation of mixes. In *EUROCRYPT*, 1989.

[143] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *ACM Conference on Computer and Communications Security*, 2000.

[144] A. Di Pierro, C. Hankin, and Herbert Wiklicky. On approximate non-inteference. In *IEEE Computer Security Foundations Workshop*. IEEE, 2002.

[145] S. Pinsky. Absorbing covers and intransitive non-interference. In *Symposium on Security and Privacy*. IEEE, 1995.

[146] B. Randell and P.Y.A. Ryan. Voting technologies and trust. *IEEE Security & Privacy*, 2005. To appear.

[147] M.G. Reed, P.F. Syverson, and D.M. Goldschlag. Anonymous connections and onion routing. *IEEE Selected Areas in Communication Special Issue on Copyright and Privacy Protection.* 1998.

[148] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *Journal of Computer Security*, 1(1), 2004.

[149] W. Riesig and G. Rozenberg, editors. *Lectures on Petri Nets.* Number 1491 & 1492 in Lecture Notes in Computer Science. Springer-Verlag, 1998.

[150] A. Roscoe. CSP and determinism in security modelling. In *Symposium on Security and Privacy.* IEEE, 1995.

[151] A. Roscoe, R. Forster, and G. Reid. Successes and failures of behavioural models. In *Millennial Perspectives in Computer Science.* Palgrave, 2000.

[152] A. Roscoe and M. Goldsmith. What is intransitive non-interference? In *Symposium on Security and Privacy.* IEEE, 1999.

[153] A. Roscoe, Jim Woodcock, and L. Wulf. Non-interference through determinism. *Journal of Computer Security*, 4(1):27–54, 1994.

[154] W. Roscoe. *The Theory and Practice of Concurrency.* Prentice Hall, 1998.

[155] J. Rushby. Design and verification of secure systems. In *Symposium on Operating Systems Principles.* ACM, 1981.

[156] J. Rushby. Non-interference, transitivity and channel security properties. Technical Report CSL-92-02, SRI International, 1992.

[157] P. Y. A. Ryan. A CSP formulation of non-interference and unwinding. In *IEEE Computer Security Foundations Workshop.* IEEE, 1990.

[158] P. Y. A. Ryan. A CSP formulation of non-interference. *Cipher - Newsletter of the IEEE Technical Committee on Security & Privacy*, Winter:19–27, 1991.

[159] P.Y.A. Ryan. Putting the human back in voting protocols. In *Fourteenth International Workshop on Security Protocols*, Lecture Notes in Computer Science. Springer-Verlag. To appear.

[160] P.Y.A. Ryan. Mathematical models of computer security. In *Foundations of Security Analysis and Design*, number 2171 in Lecture Notes in Computer Science, 2001.

[161] P.Y.A. Ryan. Enforcing the unenforceable. In *International Workshop on Security Protocols*, 2003.

[162] P.Y.A. Ryan. Towards a dependability case for the Chaum voting scheme, 2004. DIMACS Workshop on Electronic Voting – Theory and Practice.

[163] P.Y.A. Ryan and T. Peacock. Prêt à voter: a systems perspective. Technical Report CS-TR-929, University of Newcastle upon Tyne, 2005.

[164] P.Y.A. Ryan and S. Schneider. Process algebra and non-interference. *Journal of Computer Security*, 9(1-2):75–103, 2001.

[165] P.Y.A. Ryan and S. A. Schneider. Prêt à voter with re-encryption mixes. Technical Report CS-TR-956, University of Newcastle upon Tyne, 2006.

[166] A. Sabelfeld and A. Myers. Language-based information flow security. *IEEE Journal on Selected Areas in Communications*, 21(1), 2003.

[167] A. Sabelfeld and D. Sands. Dimensions and principles of declassification,. In *IEEE Computer Security Foundations Workshop*, 2005.

[168] K. Sako and J. Killian. Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In *Advances in Cryptology*, number 921 in Lecture Notes in Computer Science, pages 393–403. Springer-Verlag, 1995.

[169] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems,. In *Proceedings of the IEEE*, volume 63, pages 1278–1308, 1975.

[170] P. Samarati and S. De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In *Foundations of Security Analysis and Design*, number 2171 in Lecture Notes in Computer Science, 2001.

[171] L. Sassaman, B. Cohen, and N. Mathewson. The Pynchon Gate: A secure method of pseudonymous mail retrieval. In *Workshop on Privacy in the Electronic Society*, 2005.

[172] F. B. Schneider. Enforceable security properties. *ACM Transactions on Information and System Security*, 3(1), 2000.

[173] S. Schneider. Security properties and CSP. In *Symposium on Security and Privacy*. IEEE, 1996.

[174] S. Schneider. *Concurrent and Real-time Systems: the CSP Approach*. Wiley. 1999.

[175] S. Schneider. May testing, non-interference, and compositionality. *Electronic Notes in Theoretical Computer Science*, 40, 2001.

[176] S. Schneider and A. Sidiropoulos. CSP and anonymity. In *European Symposium on Research in Computer Security*, number 1146 in Lecture Notes in Computer Science, 1996.

[177] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 & 623–656, July & Oct. 2002.

[178] D. Sutherland. A model of information. In *9th National Security Conference*, 1986.

[179] I. Sutherland, S. Perlo, and V. Varadharajan. Deddicibility security with dynamic level assignments. In *IEEE Computer Security Foundations Workshop*. IEEE, 1989.

[180] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5), 2002.

[181] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5), 2002.

[182] P.F. Syverson and S. G. Stubblebine. Group principals and the formalization of anonymity. In *World Congress on Formal Methods*, number 1708 in Lecture Notes in Computer Science. Springer-Verlag, 1999.

[183] P.F. Syverson and S. G. Stubblebine. Authentic attributes with fine-grained anonymity protection. In *Financial Cryptography*, 2001.

[184] V. Varadharajan. Petri net modelling of information flow security requirements. In *IEEE Computer Security Foundations Workshop*. IEEE, 1990.

[185] D. von Oheimb. Information flow revisited: Noninfluence = noninterference + nonleakage. In *European Symposium on Research in Computer Security*, 2004.

[186] J. Wittbold and D. Johnson. Information flow in nondeterministic systems. In *Symposium on Security and Privacy*. IEEE, 1990.

[187] A. Zakinthinos and E. S. Lee. A general theory of security properties. In *Symposium on Security and Privacy*. IEEE, 1977.

[188] S. Zdancewic and A. Myers. Robust declassification. In *IEEE Computer Security Foundations Workshop*. IEEE, 2001.