Electronic Communications of the EASST
Volume 66 (2013)

Proceedings of the Automated Verification of Critical Systems (AVoCS 2013)

Automated Analysis of Voting Systems with Dolev-Yao Intruder Model

Murat Moran and James Heather

15 pages

Guest Editors: Steve Schneider, Helen Treharne
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Automated Analysis of Voting Systems with Dolev-Yao Intruder Model

Murat Moran[1*] **and James Heather**[2]

[1] m.moran@surrey.ac.uk, [2] j.heather@surrey.ac.uk
http://www2.surrey.ac.uk
Department of Computing
University of Surrey, UK

Abstract: This paper presents a novel intruder model for automated reasoning about anonymity properties of voting systems. We adapt the lazy spy for this purpose, as it avoids the eagerness of pre-computation of unnecessary deductions, reducing the required state space for the analysis. This powerful intruder behaves as a Dolev-Yao intruder, which not only observes a protocol run, but also interacts with the protocol participants, overhears communication channels, intercepts and spoofs any messages that he has learned or generated from any prior knowledge.

We make several important modifications in relation to existing channel types and the deductive system. For the former, we define various channel types for different threat models. For the latter, we construct a large deductive system over the space of messages transmitted in the voting system model.

The model represents the first formal treatment of the vVote system, which is planned for use in 2014 in state elections in Victoria, Australia.

Keywords: Lazy spy, Dolev-Yao, Voting Systems, Model Checking, CSP, FDR

1 Introduction

This paper presents a novel intruder model for automated reasoning about anonymity and secrecy properties of voting systems. It is much stronger than the passive attacker used in previous work in [MHS12, MHS13], as it behaves as a Dolev-Yao intruder [DY83]. This type of intruder not only observes a protocol run, but also interacts with the protocol participants, overhears communication channels, intercepts and spoofs any messages that he has learned or inferred from previous knowledge. This approach is inspired by lazy spy (perfect spy) [RG97], which is designed for cryptographic protocol analysis, and called 'lazy' as it avoids the eagerness of pre-computation of unnecessary inferences. To apply this intruder model to voting systems, several important modifications are needed in relation to existing channel types and the deductive system. For the former, we benefit from Creese *et al.* [CGRZ03, CGH^{+}05], who defined various channels for different threat models in ubiquitous computing environments. For the latter, we construct a larger deductive system over the space of messages transmitted in the model.

The basis for our model is the vVote voting system, which is based on Prêt à Voter ('PaV') [Rya04], and is under development for use in Victorian Electoral Commission (VEC) elections

[*] This author is sponsored by the Ministry of Education Republic of Turkey

in 2014 [BCH+12a, BCH+12b, Cul13] in Victoria, Australia. These elections typically take over three million votes, electing 88 legislative assembly and 40 legislative council representatives, using a mixture of the alternative vote[1] (AV), and the single transferable vote (STV). Most of the key features of PaV are retained in the vVote system. However, to adapt the system to such a complex election setup, a number of modifications have been necessary in the system design; for instance, the inclusion of distributed ballot generation, an electronic ballot marker (EBM) to assist the voter in filling out the ballot, and print-on-demand ballots for voters who are voting away from their registered polling station.

In the literature, there has to date been no successful automated anonymity verification of voting systems using the Dolev-Yao intruder model. For example, Backes *et al.* [BHM08] analysed voting systems mechanically in terms of verifiability properties. However, no automated analysis of anonymity property was provided as the ProVerif tool employed was unable to cope with algebraic equivalences, and hence, a hand proof was given. Similarly, Delaune *et al.* [DRS08, DKR10] and Smyth [Smy11] verified vote privacy of the FOO voting system with an additional compiler (ProSwapper), but these lacked a proof of its soundness — we understand this as that the framework may produce false negatives. Chadha *et al.* [CCK12] managed to verify the anonymity of the FOO voting system using a prototype, Active Knowledge in Security Protocols (AKISS), which was written in the OCaml programming language and implemented to check equivalences; however, the tool used was inefficient, and an important part of the analysis, the termination of the saturation procedure as required for deciding trace equivalences, was merely conjectured rather than proven.

This paper makes two important contributions. First, we adapt the lazy spy intruder model to voting system analysis, as it is very efficient in terms of cutting down unnecessary states as well as being flexible for usage with other privacy-related properties. Secondly, in order to demonstrate the suitability of this intruder model for evaluating voting systems, we model and analyse a real-world voting system that is set to be employed on a large scale next year.

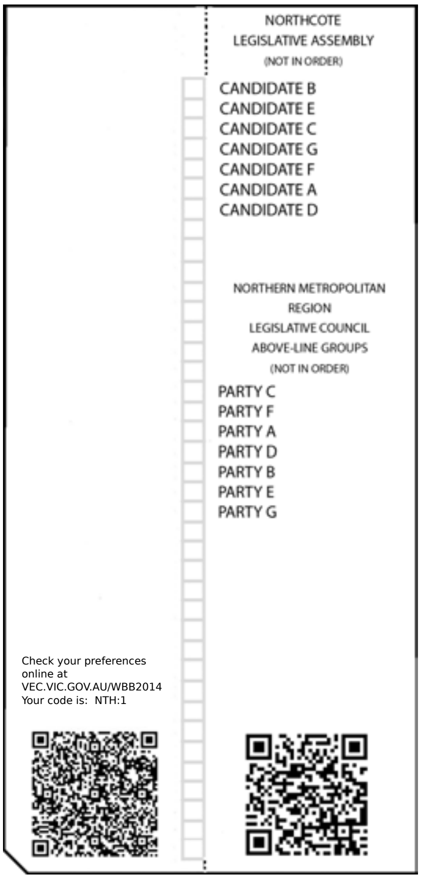
The rest of the paper is structured as follows. Section 2 presents an overview of the vVote voting system. In Section 3, the vVote voting system is modelled in CSP and the lazy spy intruder model is further extended for the analysis of voting systems. Section 4 analyses the system model regarding the formal specification of anonymity given in [MHS12], and then investigates the analysis of the model under alternative assumptions, such as the presence of a corrupt election authority. Section 5 presents conclusions and discusses the findings.

2 vVote System Outline

Over the last few decades many trustworthy voting systems have been proposed. However, only a few have been deployed in large-scale real elections. The vVote voting system, to be used in state elections in Victoria, Australia, is a paper-based electronic voting system based on PaV [Rya04]. However, a number of modifications have been made to the original PaV system. The main difference is the electronic ballot marker (EBM) deployed in order to facilitate accommodating a candidate list with over 30 candidates on the ballot forms. This also helps voters to indicate their preferences among many other candidates.

[1] also called instant-runoff voting (IRV)

The vVote ballot form illustrated in Figure 1 is similar to a PaV one. On one side there is a randomly permuted candidate list and a QR code at the bottom that records the permuted candidate order; on the other are marking boxes, a unique serial number and another QR code, corresponding to the cryptographic value that embeds the candidate order. The ballot form can be split into two, down the middle, and the right-hand side shredded; the remaining left-hand side then represents an encrypted vote, in the sense that without knowledge of the random candidate ordering it reveals nothing about the content of the vote.



NORTHCOTE
LEGISLATIVE ASSEMBLY
(NOT IN ORDER)

□ CANDIDATE B
□ CANDIDATE E
□ CANDIDATE C
□ CANDIDATE G
□ CANDIDATE F
□ CANDIDATE A
□ CANDIDATE D

NORTHERN METROPOLITAN
REGION
LEGISLATIVE COUNCIL
ABOVE-LINE GROUPS
(NOT IN ORDER)

□ PARTY C
□ PARTY F
□ PARTY A
□ PARTY D
□ PARTY B
□ PARTY E
□ PARTY G

Check your preferences
online at
VEC.VIC.GOV.AU/WBB2014
Your code is: NTH:1

Figure 1: vVote ballot form [Cul13]

2.1 Election Phases

We now explain the election phases covering the voting ceremony, and the vVote system components.

Pre-election The pre-election phase covers the preparation of election material before the polling station opens. In this period, digital ballots are generated in a distributed fashion that are encrypted under the print-on-demand (POD) service's and the election authority's public key, before being committed to the public bulletin board. Additionally, mixnets are set up, and key generation is performed in this phase.

Vote Casting This phase starts with polling stations opening and lasts until the election is closed, with no further votes being allowed to be cast. The POD service allocates and transfers pre-prepared digital ballots to a print station in the polling booth during the election. Once the voter has registered with the poll worker, the voter or the poll worker interacts with the POD service to get a ballot paper as illustrated in Figure 1. The POD client will derive the permuted candidate list on the ballot form when it is actually being printed in the polling station. The print-out also contains a QR code that contains a serial number and the candidate ordering. The voter scans this barcode into the EBM, and can now see and cast her ballot form electronically. The EBM is a new front-end component that interacts with the web bulletin board (WBB) to submit the vote and receive a digitally signed receipt for it, which is then printed for the voter for verification purposes by a receipt printer in the booth. The WBB commits, records and broadcasts the ballot data generated during the election, and also signs the serial numbers allocated by the ballot manager; this signature is included in the voter's receipt. Once the voter has cast her vote and received her receipt, she then leaves the polling station.

Post Election Post election is the phase where the cast votes are mixed by the mixnets, decrypted and tallied by a set of key sharers, in such a way that only a threshold set of these sharers can perform the decryption. The results are then announced on the bulletin board.

2.2 Ballot Generation

Ballot generation can be realised on the machine that prints the ballot form in the booth or in a distributed fashion, i.e. a number of candidate list mixers shuffle the encrypted candidate names for each vote, which ensures that the candidate ordering is random and not generated by a single party. In the distributed version, a list of encrypted ballots, including a serial number, the cryptographic value encoding the candidate list, and the list of encrypted candidate names for the printer with a proof of correspondence is produced. The printers' (POD client's) candidate list is encrypted under a threshold key shared across a set of candidate list key sharers, called the Print-on-Demand (POD) service. Hence, in order for a printer to obtain the candidate list, it generates a blinding factor, encrypts it under the POD service public key, and sends it to the POD service with a proof of knowledge. Afterwards, having received the encrypted candidate list blinded by itself, the printer removes the blinding factor, and prints the candidate list.

2.3 vVote POD Service and Protocol

The POD service provides distribution of digital ballots in a distributed manner to the polling stations in any district. As the digital ballots are prepared and committed to the WBB before the election, this service facilitates the print-on-demand ballot distribution in real time. (The details about the POD service and any other part of the vVote system can be found in the software design technical report [Cul13]. The core design is stable, though minor design aspects might change before the system goes live.)

In the ballot generation procedure, the randomised candidate order of a ballot is encrypted under the election public key pk_{EA} and it is then transformed to an encryption under the POD service's public key pk_{PS} without revealing the underlying message, as described in [Jak99]. The same transformation technique is also used in the POD protocol to transform the encryptions on the digital ballots into the designated POD client's public key pk_{PC} and these transformed ciphertexts cannot be decrypted by anyone other than the designated printer.

In this new system, the electronic ballot marker (EBM) is particularly interesting as it forms the key distinctive characteristic of vVote, being the only device in the system that knows how a particular voter has voted. That is, when the voter transfers her actual ballot form to the EBM, the candidate list on her form is also transferred to the EBM, while it is destroyed and kept secret in traditional PaV. One of the assumptions made in [BCH$^+$12a, BCH$^+$12b, Cul13] is that the POD client, where physical ballot form is printed, and the EBM, are located in a private environment, such as a voting booth.

3 vVote and Intruder Model

Security protocols consist of two or more agents sending messages to each other via a shared medium or on direct communication channels. The vVote voting system is modelled in terms of a number of agent processes that run in parallel, each modelling a different component of the system. Although the aim here is to obtain a model that reflects real system behaviour as closely as possible, there are a few assumptions that need to be made in order to avoid state explosion, which also result in abstractions in some of the features of this voting system. For instance, although vVote supports the AV and STV electoral methods, we will here model only plurality voting, or first-past-the-post (FPTP). Additionally, in the original vVote system, ballot generation is performed in a distributed fashion, and similarly the bulletin board is a threshold-based service, which signs messages by co-operation, whereas in the model these threshold parties are treated as single entities. The purpose of the thresholding is to ensure that the service guarantees correct behaviour as long as some threshold number of the servers remain honest; the appropriate abstraction here is thus to model the distributed service as a single trusted entity, whose behaviour corresponds to the overall behaviour of the distributed service provided that the threshold assumption is met.

3.1 Datatypes, Messages and Channels

We follow the typical formal approach to modelling of security protocols, and treat cryptographic primitives, such as encryptions and signatures, as symbolic operations with the appropriate algebraic properties—for instance, public key encryption: $\mathsf{E}_{pk}(f)$, decryption: $\mathsf{D}_{sk}(f)$, signature: $\mathsf{S}_{sk}(f)$, where *pk* and *sk* are the corresponding public secret keys, respectively. A message including a serial number s, and an encrypted candidate list l (called raw ballots here) is denoted as $\mathsf{Raw}(s, \mathsf{E}_{pk}(l))$, and a digital ballot message formed by a signed serial number and an encrypted candidate list is modelled as $\mathsf{DigB}(\mathsf{S}_{sk}(s), \mathsf{E}_{pk}(l))$. Similarly, a ballot form consisting of a candidate list, a serial number, and an index value $\mathsf{Ind}.i$, is $\mathsf{B}(l, s, \mathsf{Ind}.i)$; a message with a serial number and an index value forming the marking boxes on the ballot form, demonstrated as $\mathsf{RHS}(s, \mathsf{Ind}.i)$; and a receipt is denoted as $\mathsf{R}(\mathsf{S}_{sk}(\mathsf{RHS}(s, \mathsf{Ind}.i)))$. Finally, a message consisting of an index value and an encrypted candidate list is called a *vote* and shown as $\mathsf{V}(\mathsf{Ind}.i, \mathsf{E}_{pk}(l))$.

In order to compose these messages, the model consists of several finite sets of facts, $\mathcal{F}$, as listed below. The abbreviation W stands for the web bulletin board, T is Tom, the poll worker, *EA* is the election authority, *PS* and *PC* are the print-on-demand service and client respectively, and *BM* is the ballot manager. For convenience, names are abbreviated as follows: the set of candidates as $\mathcal{C}$, voters as $\mathcal{V}$, agents as $\mathcal{A}$, serial numbers as $\mathcal{S}$, nonces as $\mathcal{N}$, the set of all possible candidate lists as $\mathcal{L}$, the set of indices as $\mathcal{I}$, and public keys and secret keys as $\mathcal{PK}$ and $\mathcal{SK}$ respectively.

$$\mathcal{C} = \{Zoe, Victor\} \quad \mathcal{V} = \{Alice, Bob, James\} \quad \mathcal{S} = \{s_1, s_2, s_3\} \quad \mathcal{N} = \{n_a, n_b, n_c\}$$
$$\mathcal{PK} = \{pk_W, pk_T, pk_{PS}, pk_{PC}, pk_{BM}, pk_{EA}\} \quad \mathcal{SK} = \{sk_W, sk_T, sk_{PS}, sk_{PC}, sk_{BM}, sk_{EA}\}$$
$$\mathcal{A} = \bigcup(\mathcal{V}, \{Tom, authority, wbb, teller, podservice, podclient, ballotmngr, ebm, printer\})$$

The agents send various kinds of messages to each other, which need to be defined in terms of datatypes. The messages mentioned above form the message set $\mathcal{M}$.

The framework used in [Ros97, RSG+00] involves only insecure communication channels, and hence, any message can be manipulated in many ways by the intruder. Such an assumption is too strong for voting systems that require an environment for the voters to be able to vote privately, such as a voting booth, at least if the action of receiving a ballot form is modelled as a message. This is also the case for most of the remote voting systems, where it is assumed that no one is watching over the voter's shoulder while she is marking her ballot paper. Hence, this necessitates the existence of private channels in the voting system model. To this end, the agents in the model are enabled to communicate over a secure channel, called *scomm*, on which the intruder has no power at all. In addition, from the observations made throughout the analysis (which will be explained further in Section 5), it is assumed that at least two eligible honest voters are able to vote, and the cast votes are tallied at the end of the election. This assumption requires that there is a channel in the voting system model such that the communications made by these two honest voters with the other agents are *No Spoofing and Blocking* (NSB) channels modelled as *nsbcomm* here. On such channels the intruder can overhear the communication, but cannot block its occurrence and spoof any messages. Creese *et al.* [CGRZ03, CGH^{+}05] describe various kinds of channels for pervasive computing environments. The insecure communication channels are not expressed as a proper channel type here, but instead is modelled as a special power given to the intruder in his CSP process definition, where he can overhear, block and spoof messages. (It could equally be modelled such that all channels are insecure, and the NSB and private channels are the subset of these insecure channels, whereby the intruder would be restricted to define what he is capable of doing as in [CGRZ03, CGH^{+}05]. However, in this modelling, it is more convenient, first, to model the private and NSB channels, restricting the intruder and then give him more power to model the insecure communications. That is, instead of defining what he cannot block or spoof in a model, here, what he can do in the system is specified.)

The channels have the form $\mathcal{A}.\mathcal{A}.\mathcal{M}$, where $\mathcal{A}$ is the set of agents and $\mathcal{M}$ is the set of messages that agents may transmit over the channels. For example, the private channel on which Alice sends a sensitive message m_1 (possibly scanning the ballot form) to the electronic ballot marker (EBM) is modelled as $scomm.Alice.ebm.m_1$. Alternatively, $nsbcomm.podclient.Bob.m_2$ is an example for the insecure NSB channels on which the printer prints a ballot form for Bob.

It is also important to define the set of messages that make sense to the protocol (they are from real communications between agents), called *comms*, which is defined as the union of sets of data objects for each message type. For instance, the following defines the vote messages sent by one agent to another.

$$commVotes = \{a.b.m \mid m \leftarrow votes, a \leftarrow \mathcal{A}, b \leftarrow \mathcal{A}, a \neq b\}$$

These are useful when the intruder is afforded the ability to modify the messages on the insecure channels or not to block and fake certain data from specific agents as it may be confusing as to whether the message is already known or has just been learned from the real communication that the intruder overhears.

3.2 Honest Participants

The vVote model developed for this work is defined by the processes illustrated at the top of Figure 2. All the processes are involved in the protocol by sending and receiving messages on the synchronised channels. For brevity, we give the full CSP only of the voter process here, but see [Mor13] for the full CSP$_M$ code of the vVote model and analysis, and sanity checks.

The parameterised process $Voter(v,c)$ models a voter $v \in \mathcal{V}$ voting for a chosen candidate $c \in \mathcal{C}$. There are two honest voters, Alice and Bob, and a misbehaving one, James, who behaves honestly in the model at first, but his secret will be shared with the intruder later on, and whose

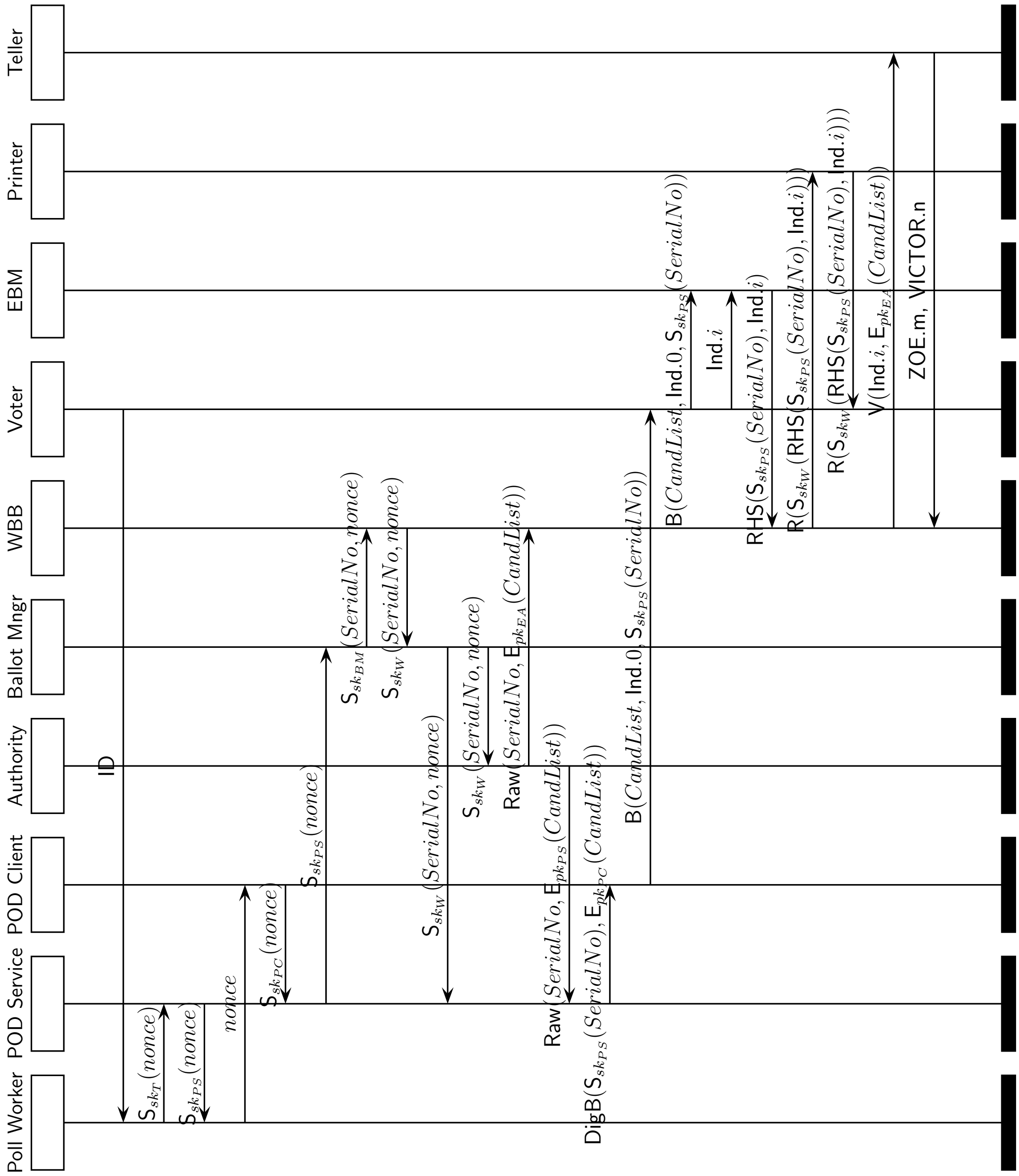

Poll Worker
POD Service
POD Client
Authority
Ballot Mngr
WBB
Voter
EBM
Printer
Teller
ID
$\mathsf{S}_{sk_T}(nonce)$
$\mathsf{S}_{sk_{PS}}(nonce)$
$nonce$
$\mathsf{S}_{sk_{PC}}(nonce)$
$\mathsf{S}_{sk_{PS}}(nonce)$
$\mathsf{S}_{sk_{BM}}(SerialNo, nonce)$
$\mathsf{S}_{sk_W}(SerialNo, nonce)$
$\mathsf{S}_{sk_W}(SerialNo, nonce)$
$\mathsf{S}_{sk_W}(SerialNo, nonce)$
$\mathsf{Raw}(SerialNo, \mathsf{E}_{pk_{EA}}(CandList))$
$\mathsf{Raw}(SerialNo, \mathsf{E}_{pk_{PS}}(CandList))$
$\mathsf{DigB}(\mathsf{S}_{sk_{PS}}(SerialNo), \mathsf{E}_{pk_{PC}}(CandList))$
$\mathsf{B}(CandList, \mathsf{Ind}.0, \mathsf{S}_{sk_{PS}}(SerialNo))$
$\mathsf{B}(CandList, \mathsf{Ind}.0, \mathsf{S}_{sk_{PS}}(SerialNo))$
$\mathsf{Ind}.i$
$\mathsf{RHS}(\mathsf{S}_{sk_{PS}}(SerialNo), \mathsf{Ind}.i)$
$\mathsf{R}(\mathsf{S}_{sk_W}(\mathsf{RHS}(\mathsf{S}_{sk_{PS}}(SerialNo), \mathsf{Ind}.i)))$
$\mathsf{R}(\mathsf{S}_{sk_W}(\mathsf{RHS}(\mathsf{S}_{sk_{PS}}(SerialNo), \mathsf{Ind}.i)))$
$\mathsf{V}(\mathsf{Ind}.i, \mathsf{E}_{pk_{EA}}(CandList))$
ZOE.m, VICTOR.n

Figure 2: vVote system model

communications, even the private and NSB ones, are used by the intruder. Once a voter has authenticated herself on the NSB channel with Tom, the poll worker, the voter receives a ballot form from the POD client with the candidate list printed on it and scans her ballot data to the EBM on the secure channel, where she can see her ballot in an electronic environment. After indicating her preference by sending the index value ($\mathsf{Ind}.i$) to the EBM that corresponds to the candidate she wants to vote for — the index i is found by using the function $\mathsf{find}(c,l)$, which finds the candidate c in the sequence of candidates, l — , she then receives her signed receipt and leaves the polling station.

All eligible voters, Alice, Bob and James, follow this protocol, which is modelled as the interleaving of all individual voter processes, $\mathit{Voters} \mathrel{\widehat{=}} \|_{v,c} \mathit{Voter}(v,c)$.

$$
\begin{aligned}
&\mathit{Voter}(v,c) \mathrel{\widehat{=}} \mathit{nsbcomm.v.Tom.v} \rightarrow \\
&\qquad \mathop{\Box}_{\substack{l \in \mathcal{L} \\ s \in \mathcal{S}}} \left(\begin{array}{l}
\mathit{scomm.podclient.v}.\mathsf{B}(l, \mathsf{S}_{sk_{PS}}(s), \mathsf{Ind}.0) \rightarrow \\
\mathit{scomm.v.ebm}.\mathsf{B}(l, \mathsf{S}_{sk_{PS}}(s), \mathsf{Ind}.0) \rightarrow \\
\mathop{\Box}_{i:=\mathsf{find}(c,l)} \left(\begin{array}{l}
\mathit{nsbcomm.v.ebm}.\mathsf{Ind}.i \rightarrow \\
\mathit{nsbcomm.printer.v}.\mathsf{R}(\mathsf{S}_{sk_W}(\mathsf{RHS}(\mathsf{S}_{sk_{PS}}(s), \mathsf{Ind}.i))) \rightarrow \\
\mathit{STOP}
\end{array} \right)
\end{array} \right)
\end{aligned}
$$

3.3 Adapting Lazy Spy

Lazy spy [RG97] is an efficient CSP implementation of the Dolev-Yao intruder model as it avoids state explosion by following only its findings (deductions through the messages he has seen or from his initial knowledge). This intruder model provides active attacks against the system by not only observing the communication channels, but also blocking messages or generating and sending fake messages to any agents on the system. We show here how to alter the model so it can work with cryptographic voting systems. In particular, the vVote voting system model is equipped with a number of system-specific messages as well as the cryptographic ones, and this necessitates defining further deduction rules so that the intruder can process these messages appropriately. Secondly, the initial knowledge of the intruder $\mathcal{IK}$ is also model-specific, and needs to be defined according to the voting system model. Lastly, because of the introduction of various channel types in the analysis of voting systems, the intruder model needs to be amended so that the private channels stay private and NSB channels are not blocked or spoofed by the intruder.

In order to allow the intruder to compose messages, there are a number of deduction rules. A deduction is a pair (X, f), where X is a finite set of facts and f is the fact that can be learnt, provided that the intruder possesses X; if the intruder is able to make this deduction then we write '$X \vdash f$'. The deduction rules regarding this analysis $\mathcal{D}$ are defined in Table 1 (the deduction rules are system-specific and cover all legitimate messages sent over the network, and inconsistent messages cannot be sent to an agent). The new deduction rule BALLOT-COMP enables ballot forms to be composed if the intruder possesses the set $\{l, \mathsf{S}_{sk}(s), \mathsf{Ind}.i\}$, where l is the candidate list, s is serial number and $\mathsf{Ind}.i$ is the index value, corresponding to the chosen candidate; conversely, the deduction rule BALLOT-DCMP helps the intruder to decompose ballot forms and obtain all the data on it. Similarly, the intruder can also work on any composition and decomposition of any other messages in the model. For instance, RHS-COMP and RHS-DCMP are the deduction rules related to cast ballot forms, consisting of an index value and a signed serial number $\{\mathsf{Ind}.i, \mathsf{S}_{sk}(s)\}$. VOTE-COMP and VOTE-DCMP are the deduction rules related to the votes, in the form of $\mathsf{V}(\mathsf{Ind}.i, \mathsf{E}_{pk}(l))$ (note that these do not contain a serial number). The deduction rules regarding the digital ballots, consisting of a signed serial number and an encrypted candidate list, $\{\mathsf{S}_{sk}(s), \mathsf{E}_{pk}(l)\}$, are DIG.BLT-COMP and DIG.BLT-DCMP. Similarly, RAW.BLT-COMP and RAW.BLT-DCMP are the two deduction rules that help the intruder compose and decompose the raw ballots, $\{s, \mathsf{E}_{pk}(l)\}$, and IND-COMP and IND-DCMP are the index-related rules.

The set *comms* needs to be defined for all messages in $\mathcal{M}$ so that the intruder can justify that a message being heard is actually from a real communication between agents. As in the protocol, no agent ever sends any message to himself.

SYM-ENC.	$\{k, m\}$	$\vdash$	$\mathsf{E}_k(m)$
SYM-DEC.	$\{k, \mathsf{E}_k(m)\}$	$\vdash$	m
ASYM-ENC.	$\{pk, m\}$	$\vdash$	$\mathsf{E}_{pk}(m)$
ASYM-DEC.	$\{sk, \mathsf{E}_{pk}(m)\}$	$\vdash$	m
SIGN-SIG.	$\{sk, m\}$	$\vdash$	$\mathsf{S}_{sk}(m)$
SIGN-EXT.	$\{pk, \mathsf{S}_{sk}(m)\}$	$\vdash$	m
BALLOT-COMP.	$\{l, \mathsf{S}_{sk}(s), \mathsf{Ind}.i\}$	$\vdash$	$\mathsf{B}(l, \mathsf{S}_{sk}(s), \mathsf{Ind}.i)$
BALLOT-DCMP.	$\{\mathsf{B}(l, \mathsf{S}_{sk}(s), \mathsf{Ind}.i)\}$	$\vdash$	$l, \mathsf{S}_{sk}(s), \mathsf{Ind}.i$
RHS-COMP.	$\{\mathsf{Ind}.i, \mathsf{S}_{sk}(s)\}$	$\vdash$	$\mathsf{RHS}(\mathsf{Ind}.i, \mathsf{S}_{sk}(s))$
RHS-DCMP.	$\{\mathsf{RHS}(\mathsf{S}_{sk}(s), \mathsf{Ind}.i)\}$	$\vdash$	$\mathsf{Ind}.i, \mathsf{S}_{sk}(s)$
VOTE-COMP.	$\{\mathsf{Ind}.i, \mathsf{E}_{pk}(l)\}$	$\vdash$	$\mathsf{V}(\mathsf{Ind}.i, \mathsf{E}_{pk}(l))$
VOTE-DCMP.	$\{\mathsf{V}(\mathsf{Ind}.i, \mathsf{E}_{pk}(l))\}$	$\vdash$	$\mathsf{Ind}.i, \mathsf{E}_{pk}(l)$
DIG.BLT-COMP.	$\{\mathsf{S}_{sk}(s), \mathsf{E}_{pk}(l)\}$	$\vdash$	$\mathsf{DigB}(\mathsf{S}_{sk}(s), \mathsf{E}_{pk}(l))$
DIG.BLT-DCMP.	$\{\mathsf{DigB}(\mathsf{S}_{sk}(s), \mathsf{E}_{pk}(l))\}$	$\vdash$	$\mathsf{S}_{sk}(s), \mathsf{E}_{pk}(l)$
RAW.BLT-COMP.	$\{s, \mathsf{E}_{pk}(l)\}$	$\vdash$	$\mathsf{Raw}(s, \mathsf{E}_{pk}(l))$
RAW.BLT-DCMP.	$\{\mathsf{Raw}(s, \mathsf{E}_{pk}(l))\}$	$\vdash$	$s, \mathsf{E}_{pk}(l)$
IND-COMP.	$\{i\}$	$\vdash$	$\mathsf{Ind}.i$
IND-DCMP.	$\{\mathsf{Ind}.i\}$	$\vdash$	i

Table 1: Deduction rules capturing the properties of cryptographic primitives and the vVote voting system messages (*sk* and *pk*).

$$comms = \{a.b.m \mid m \leftarrow \mathcal{M}, a \leftarrow \mathcal{A}, b \leftarrow \mathcal{A}, a \neq b\}$$

The messages in the model that make sense to the intruder are: *comms*, all the messages from real communications; *Nsbcomms*, the set of messages that can be overheard but not blocked or spoofed; and *Ucomms*, the set of insecure messages for which he can act as a Dolev-Yao intruder.

As all honest participants communicate on the NSB channels, not including a message type in *Nsbcomms* means that the intruder cannot even overhear that kind of message and this is how we introduce the private channels. Hence, the messages in the form of a ballot are not included in this set, as the intruder should not be able to observe any communication involving a ballot form between honest participants (denoted as *commBallots*); in fact, we have $Nsbcomms = comms \setminus commBallots$ — $commBallots$ forms the private channels. Additionally, the insecure messages that the intruder can overhear, block or use in any way in the line of Dolev-Yao model, are defined with the set *Ucomms* as follows. It should be noted that the set in the analysis of vVote covers all the messages that are communicated by the dishonest voter James.

$$\begin{aligned} Ucomms = \bigcup(&\{a.a'.f \mid a.a'.f \leftarrow comms, a \leftarrow \{James\}, a' \leftarrow agents\},\\ &\{a.a'.f \mid a.a'.f \leftarrow comms, a \leftarrow agents, a' \leftarrow \{James\}\}) \end{aligned}$$

The adaptation of the intruder model to voting systems analysis has been made by introducing different channel types, introduced in this section, and the CSP definition of the lazy spy intruder model is kept intact. Hence, the lazy spy intruder model, which is called *Intruder* here and defined in terms of the channels *learn* and *say*, is not covered in detail in this paper, but further details can be found in [Ros97].

3.4 Putting the Network Together

Figure 3 illustrates how the intruder is connected to the dishonest voter James and the honest voter Alice: Alice's private channel *scomm* is kept private, but her insecure NSB channels can be observed by the intruder, whereas all the channels of James are under the control of the intruder.

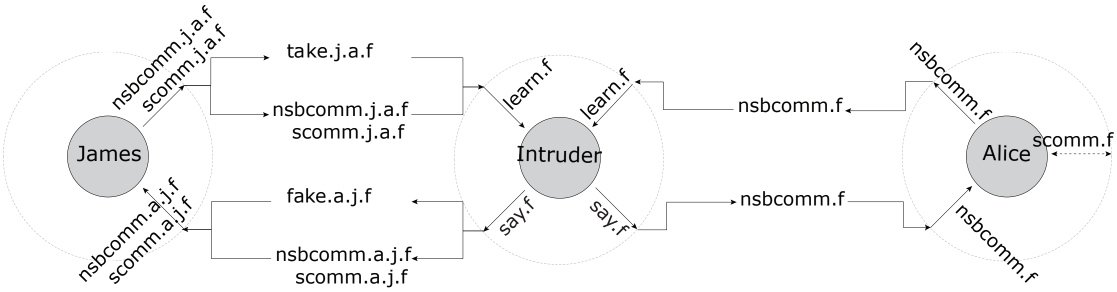


Figure 3: Intruder's communication channels with the dishonest voter James and honest Alice.

The processes that construct the voting system model and the intruder model are connected by using the renaming operator. That is, *nsbcomm.a.b.m* and *learn.m* channels are renamed to a *take.a.b.m* channel, and the *nsbcomm.b.a.m* and *say.m* channels are renamed to a *fake.b.a.* channel from the agent *a*'s point of view. Similarly, the intruder process is also renamed and the aim is to connect them as is done in Figure 3. To this end, a renaming function *r* for the process *P* and agent name *p* can be defined as follows:

$$\begin{aligned}r(P,p) \mathrel{\widehat{=}} P&[\![{}^{nsbcomm.p,take.p}/_{nsbcomm.p,nsbcomm.p}]\!]\\&[\![{}^{nsbcomm.a.p,fake.a.p}/_{nsbcomm.a.p,nsbcomm.a.p} \mid a \in \mathcal{A}]\!]\\&[\![{}^{scomm.p,take.p}/_{scomm.p,scomm.p}]\!]\\&[\![{}^{scomm.a.p,fake.a.p}/_{scomm.a.p,scomm.a.p} \mid a \in \mathcal{A}]\!]\end{aligned}$$

The renamed voter process for the voter v, for instance, can be defined with $r(Voter(v,c),v)$. Note that, the private channel *scomm* is renamed to the *take* and *fake* channels, because this models the malicious behaviour of a corrupt voter, and secondly it may seem like that the intruder can take and fake private channels at the moment, but this is prevented later on when we define renamed intruder. Similarly, the other processes that construct the vVote voting system model are renamed as in the above example. Consequently, the voting system model, *Model*, which is ready to be modified by the intruder, is defined as the parallel composition of all those renamed processes.

$$\begin{aligned}Model \mathrel{\widehat{=}} &rVoters \parallel rPollworker \parallel rAuthority \parallel rEBM \parallel rPodservice \parallel rPrinter\\&\parallel rBallotmanager \parallel rPodclient \parallel rWBB \parallel rTeller\end{aligned}$$

The parallel composition above is constructed in a way that the processes only synchronise on the *nsbcomm* and *scomm* channels on which they send messages to each other, leaving the insecure channels (*take*, *fake*) vulnerable to be used by the intruder. The following shows how two processes are put in parallel; the above parallel composition of *Model* should be interpreted along these lines.

$$Model \mathrel{\widehat{=}} rVoters \underset{X}{\parallel} rPollworker \parallel \ldots$$

where $X = \{\mid nsbcomm.v.Tom, nsbcomm.Tom.v, scomm.v.Tom, scomm.Tom.v \mid v \leftarrow \mathcal{V} \mid\}$

Similarly, the intruder process is prepared by renaming as below so that the intruder can overhear the messages on the insecure NSB channels (*Nsbcomms*) and act as the Dolev-Yao intruder on the insecure channels (*Ucomms*), but keep the private channels private.

$$\begin{aligned}rIntruder \mathrel{\widehat{=}} Intruder&[\![{}^{say,learn}/_{say,say}]\!]\\&[\![{}^{nsbcomm.a.a'.f,take.b.b'.f}/_{learn.f,learn.f} \mid \begin{array}{l}a.a'.f \in Nsbcomms,\\ b.b'.f \in Ucomms,\\ a \neq a', b \neq b'\end{array}]\!]\\&[\![{}^{fake.a.a'.f}/_{say.f} \mid a.a'.f \in Ucomms,\ a \neq a']\!]\end{aligned}$$

The process $System_{\text{vVote}}$ is then defined in terms of the parallel composition of *Model* and *rIntruder*, which synchronise on the channels they share.

$$System_{\text{vVote}} \mathrel{\widehat{=}} Model \underset{\{\mid nsbcomm,take,fake \mid\}}{\parallel} rIntruder$$

4 Analysis

In this section, the first fully automated analysis of the vVote voting system is presented under a Dolev-Yao intruder model and using the anonymity definition given in [MHS12] as the specification. It requires that when the two channels $c.x$ and $d.x$ are swapped over for all values of x, if the resulting process is indistinguishable from the original one, then the process provides anonymity on those channels. To this end, we first allow the intruder to control the dishonest voter James by renaming his individual voter process as $rVoter(James, c)$ for some candidate c. However, Alice and Bob are able to vote on the NSB channels. Subsequently, for the anonymity specification, the two systems, which are expected to be indistinguishable, are defined as two separate system behaviours without using the renaming operator: on the one hand, say $System'_{vVote}$, Alice votes for Zoe, Bob votes for Victor and James can vote for either Zoe or Victor; whereas on the other hand, say $System''_{vVote}$, Alice votes for Victor, Bob votes for Zoe and again James can vote for either. The systems are modelled in such a way that the intruder can manipulate everything James does, including his private messages, whilst Bob and Alice can vote freely without any blocking or spoofing. However, the intruder can still overhear the public NSB channels.

In order to avoid false positive attacks where the intruder can distinguish two ciphertexts, even if he does not know the secret key, we deploy a masking function maskFact. The function renames all messages encrypted under a public key, whose corresponding secret key is not known by the voter, to a dummy *ciphertext*; if the secret key is in the intruder's initial knowledge, then he is allowed to differentiate two ciphertexts by not masking them.

$$\mathsf{maskFact}(\mathsf{E}_{pk}(m)) = \text{if } \mathsf{dual}(pk) \in \mathcal{IK} \text{ then } \mathsf{E}_{pk}(m) \text{ else } ciphertext$$

The masking function $\mathsf{mask}(P)$ can also be defined for the processes, which masks all encrypted facts of a given process, P, using the maskFact function for all the data that appears in this process. (No keys are ever sent over the network, so if the intruder does not know a secret key at the beginning, he will not learn it later.)

$$\begin{aligned} mask(P) \mathrel{\hat{=}} P & \llbracket ^{achannel.a.a'.\mathsf{DigB}(\mathsf{S}_{sk}(s),\mathsf{maskFact}(\mathsf{E}_{pk}(l)))} / _{achannel.a.a'.\mathsf{DigB}(\mathsf{S}_{sk}(s),\mathsf{E}_{pk}(l))} \rrbracket \\ & \llbracket ^{achannel.a.a'.\mathsf{Raw}(s,\mathsf{maskFact}(\mathsf{E}_{pk}(l)))} / _{achannel.a.a'.\mathsf{Raw}(s,\mathsf{E}_{pk}(l))} \rrbracket \\ & \llbracket ^{achannel.a.a'.\mathsf{V}(\mathsf{Ind}.i,\mathsf{maskFact}(\mathsf{E}_{pk}(l)))} / _{achannel.a.a'.\mathsf{V}(\mathsf{Ind}.i,\mathsf{E}_{pk}(l))} \rrbracket \end{aligned}$$

where $achannel \in \{nsbcomm, take, fake\}$, the serial number $s \in \mathcal{S}$, the candidate list $l \in \mathcal{L}$ and the index value $i \in \mathcal{I}$.

After applying the masking to both $System'_{vVote}$ and $System''_{vVote}$, they are ready for analysis under the anonymity specification. To this end, the anonymity requirement of this voting system model is checked with the following equivalence in which the private channels are hidden:

$$\mathsf{mask}(System'_{vVote}) \setminus \{|scomm|\} \equiv_{\mathrm{T}} \mathsf{mask}(System''_{vVote}) \setminus \{|scomm|\}$$

Failures-Divergence-Refinement (FDR) [GGH$^+$] verifies that the two systems refine each other, meaning that they are trace equivalent and hence that the intruder cannot distinguish them. We conclude that the vVote voting system model provides anonymity under the Dolev-Yao intruder model. The analysis was performed using the FDR 2.94 version on a machine with GenuineIntel CPU family 6, model 2, 1.86GHz, and 34GB RAM (4GB allocated to FDR). With the restricted Dolev-Yao model, the intruder can only block or spoof a subset (James's communications) of all messages. The restriction is modelled with the existence of private and NSB channels. Moreover, the verification of this model with three voters and two candidates takes $20m29s$ to check 16,063,214 states. However, with an additional corrupt voter, the state space explosion escalates quickly due to too much data to work on. Hence, FDR cannot handle such scenarios. Intuitively, the results scale up with the number of agents, but more thoughts are needed.

5 Further Analysis and Conclusion

Although the framework used in the previous section provides a firm foundation for analysis of voting systems, it is also important to see whether the framework supports further extensions to those assumptions made previously, because one of the important challenges in electronic voting systems may be to maintain requirements in the presence of corrupt agents. Such analyses are possible with slight modifications in this framework. The following paragraph presents an analysis of vVote in the presence of a corrupt POD service.

Corrupt POD Service The POD service is an important part of the print-on-demand protocol. It receives raw ballot data including a serial number s and candidate list encrypted under pk_{PS}, and sends the digital ballot by signing the serial number to the POD client. If the POD service is corrupt, which we model here by giving the POD service's secret key to the intruder, the raw ballot received by the POD service, say $\mathsf{Raw}(s_3, \mathsf{E}_{pk_{PS}}(\mathsf{Sq}.\langle Zoe, Victor\rangle))$, can be captured and decrypted by the intruder. Hence, the intruder can extract the candidate list $\mathsf{Sq}.\langle Zoe, Victor\rangle$ and deduce its association with the serial number s_3. Following this, when he observes that Alice's receipt with the index value $\mathsf{Ind}.1$ has the serial number s_3 on it, he is then able to infer that Alice has voted for the first candidate of the candidate list $\mathsf{Sq}.\langle Zoe, Victor\rangle$, which is Zoe. Therefore, the intruder distinguishes the two systems as Alice cannot have voted for Victor. This counter-example is produced by FDR automatically and illustrated by the following partial trace.

$\langle \ldots$
$nsbcomm.authority.wbb.\mathsf{Raw}(s_3, ciphertext),$
$nsbcomm.authority.podservice.\mathsf{Raw}(s_3, \mathsf{E}_{pk_{PS}}(\mathsf{Sq}.\langle Zoe, Victor\rangle)),$
$nsbcomm.podservice.podclient.\mathsf{DigB}(\mathsf{S}_{sk_{PS}}(s_3), ciphertext),$
$enterBooth.Alice,$
$nsbcomm.Alice.ebm.\mathsf{Ind}.1\rangle$

Although no one is supposed to be observing voter interaction with the EBM, we assumed here that the index value sent from voter to the EBM can be observed, as it will be observed anyway once she takes her receipt from the receipt printer. Thus, the two counter-examples above were found by FDR when the intruder could observe these index values. If the intruder was not allowed to do so, the counter-examples would still appear once the voter had taken her receipt.

There are numerous possible corruption scenarios that can be modelled and analysed using this framework. In particular, the one presented here emphasises the importance of the case of a corrupt entity, where the voters are at a high risk of losing their anonymity. The vVote voting system's solution to this is to have this service distributed, so that as long as the thresholded service is not corrupt overall, then the voter's anonymity will be preserved. A similar consideration applies to distributed ballot form generation. However, if the other trusted entities, like the EBM, are acting dishonestly, the system is vulnerable to various attacks.

We have also analysed the model under the full Dolev-Yao intruder model that can overhear, intercept and spoof any messages on all channels other than the private channels. From this analysis, the following counter-example was produced, which shows that with such an intruder the vVote voting system is open to anonymity attacks.

$\langle \ldots$
$scomm.podclient.Alice.\mathsf{B}(\mathsf{Sq}.\langle Zoe, Victor\rangle, \mathsf{S}_{sk_{PS}}(s_1), \mathsf{Ind}.0),$
$comm.Alice.ebm.\mathsf{Ind}.1,$
$scomm.podclient.Bob.\mathsf{B}(\mathsf{Sq}.\langle Zoe, Victor\rangle, \mathsf{S}_{sk_{PS}}(s_2), \mathsf{Ind}.0),$
$comm.Bob.ebm.\mathsf{Ind}.2,$
$closeElection,$
$comm.wbb.teller.\mathsf{V}(\mathsf{Ind}.2, ciphertext),$
$take.wbb.teller.\mathsf{V}(\mathsf{Ind}.1, ciphertext),$
$comm.teller.wbb.Zoe.0\rangle$

This tallies with the observation made in [KR05] about the FOO voting system [FOO92]. Here, the intruder blocks all the other votes except Bob's with the channel *take*. In this case, Alice has voted for Zoe with the index value Ind.1 and Bob has voted for Victor with Ind.2 on the private channels. (The candidate lists on the private channels *scomm* are hidden in the analysis and they are revealed here just for illustration.) The intruder intercepts the vote with the index value Ind.1, and waits until Bob's vote is counted. Having seen that no one has voted for Zoe, the intruder then deduces that Bob has voted for Victor. This is a genuine and generic attack, not only on vVote, but applicable to any voting system. However, as it is not likely in a real system that the intruder can block all votes but one, it was assumed in our analysis that at least two honest votes are tallied at the end of the election.

In this paper, we have proposed a formal approach to modelling and analysis of cryptographic voting systems. In order to validate the suitability of the framework, the vVote voting system was analysed against an anonymity specification. To do so, an extensive number of other such rules regarding voting systems have been defined. These enable the intruder to learn and deduce further from his knowledge so as to able to use it to break the protocol objectives. Moreover, we introduced special channel types, private and NSB channels, in order to reason about voting systems under appropriate assumptions, as it has been observed that no voting system model can provide anonymity under an unrestricted Dolev-Yao intruder model. The framework can be applied to other voting systems providing that a CSP model of the system that is compatible with the framework is produced, and system-specific deduction rules are given.

Acknowledgements. The authors would like to thank Steve Schneider for useful discussions on the technical content.

References

[BCH+12a] C. Burton, C. Culnane, J. Heather, T. Peacock, P. Y. A. Ryan, S. Schneider, S. Srinivasan, V. Teague, R. Wen, Z. Xia. A Supervised Verifiable Voting Protocol for the Victorian Electoral Commission. In *Electronic Voting*. Pp. 81–94. 2012.

[BCH+12b] C. Burton, C. Culnane, J. Heather, T. Peacock, P. Y. A. Ryan, S. Schneider, S. Srinivasan, V. Teague, R. Wen, Z. Xia. Using Prêt à Voter in Victoria State Elections. In *EVT / WOTE*. 2012.

[BHM08] M. Backes, C. Hritcu, M. Maffei. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-Calculus. In *CSF*. Pp. 195–209. 2008.

[CCK12] R. Chadha, S. Ciobaca, S. Kremer. Automated Verification of Equivalence Properties of Cryptographic Protocols. In Seidl (ed.), *Programming Languages and Systems*. LNCS 7211, pp. 108–127. Springer Berlin Heidelberg, 2012.

[CGH+05] S. Creese, M. Goldsmith, R. Harrison, B. Roscoe, P. Whittaker, I. Zakiuddin. Exploiting Empirical Engagement in Authentication Protocol Design. In *SPC*. Pp. 119–133. 2005.

[CGRZ03] S. Creese, M. Goldsmith, B. Roscoe, I. Zakiuddin. The Attacker in Ubiquitous Computing Environments: Formalising the Threat Model. In *FAoC*. 2003.

[Cul13] C. Culnane. Software Design for VEC vVote System. Technical report CS-13-01, University of Surrey, 2013.

[DKR10] S. Delaune, S. Kremer, M. Ryan. Verifying Privacy-Type Properties of Electronic Voting Protocols: A Taster. In *Towards Trustworthy Elections*. Pp. 289–309. 2010.

[DRS08] S. Delaune, M. Ryan, B. Smyth. Automatic Verification of Privacy Properties in the Applied pi Calculus. In Karabulut et al. (eds.), *Trust Management II*. IFIP - The International Federation for Information Processing 263, pp. 263–278. Springer US, 2008.

[DY83] D. Dolev, A. C. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory* 29(2):198 – 208, mar 1983.

[FOO92] A. Fujioka, T. Okamoto, K. Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *AUSCRYPT*. Pp. 244–251. 1992.

[GGH+] P. Gardiner, M. Goldsmith, J. Hulance, D. Jackson, B. Roscoe, B. Scattergood, B. Armstrong. FDR2 User Manual. http://www.fsel.com/documentation/fdr2/html/index.html.

[Jak99] M. Jakobsson. On Quorum Controlled Asymmetric Proxy Re-encryption. In *Proc. of the Second Int'l Workshop on Practice and Theory in Public Key Cryptography*. 1999.

[KR05] S. Kremer, M. Ryan. Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In *ESOP*. Pp. 186–200. 2005.

[MHS12] M. Moran, J. Heather, S. Schneider. Verifying Anonymity in Voting Systems Using CSP. *Formal Aspects of Computing*, pp. 1–36, 2012.

[MHS13] M. Moran, J. Heather, S. A. Schneider. Automated Anonymity Verification of the ThreeBallot Voting System. In *IFM*. Pp. 94–108. June 2013.

[Mor13] M. Moran. CSP codes for CVS, ThreeBallot, Prêt à Voter and vVote Voting Systems. May 2013. http://muratmoran.wordpress.com/publications/.

[RG97] A. Roscoe, M. Goldsmith. The Perfect "spy" for Model-checking Cryptoprotocols. In *DIMACS workshop on the design and formal verification of cryptographic protocols*. 1997.

[Ros97] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.

[RSG+00] P. Y. A. Ryan, S. A. Schneider, M. H. Goldsmith, G. Lowe, A. W. Roscoe. *The Modelling and Analysis of Security Protocols : the CSP Approach*. Addison-Wesley Professional, first edition, 2000.

[Rya04] P. Y. A. Ryan. A variant of the Chaum Voter-verifiable Scheme. Technical report CS-TR-864, University of Newcastle upon Tyne, 2004.

[Smy11] B. Smyth. *Formal Verification of Cryptographic Protocols with Automated Reasoning*. PhD thesis, School of Computer Science, University of Birmingham, 2011.