Formal Specification of Electronic Ballot Transmission Protocols in the Applied Pi-Calculus

Danny F. Wuysang
dfwuysang@up.edu.ph

Henry N. Adorna
hnadorna@dcs.upd.edu.ph

Algorithms and Complexity Laboratory
Department of Computer Science
University of the Philippines Diliman
Quezon City, Philippines

ABSTRACT

In this paper, we present our recent work on formally specifying transmission protocols used in electronic voting. We use the applied pi-calculus, a formal language for concurrent and communicating processes with extensions that are particularly advantageous in modelling cryptographic protocols. We checked for standard secrecy based on reachability using the tool, ProVerif. The output showed that the keys and some results are secure, while others were proven to be vulnerable. We also found that the case of corrupt administrator exist in one of the schemes, which we proceed to show formally.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*protocol verification*; D.2.4 [**Software Engineering**]: Software/Program Verification—*formal methods*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*mechanical verification, specification techniques*

General Terms

Design, Languages, Security, Theory, Verification

Keywords

electronic voting, applied pi-calculus, transmission, cryptographic protocols, reachability, secrecy properties

1. INTRODUCTION

For almost three decades[1], researchers have been working on *electronic voting (e-voting)*. However, reception in the form of large scale deployment (e.g. national election) is still a mix of positives and negatives. Several states have opted for e-voting systems (Brazil [24], India [28]), but some turned out to be unsuccessful experiments. Discovery of some irregularities led many to doubt, even some to abandon e-voting (Ireland [19, 13], Netherlands [18]). The main issue of e-voting now is *trust*. Can we trust an electronic voting system to handle such a critical process in our democracy? Or should we go back to our "trusted" way of doing it, the conventional way?

Surely, e-voting has advantages over conventional voting. The most notable property that manual schemes could not provide is *verifiability*. That is, voters can verify that their choice(s) was recorded and counted as they intended. This is one of the main goals of e-voting systems. Another significant improvement is *efficiency* with respect to time. By employing e-voting systems, we profit from the inherent speed of computers and its network. [17] contains a list of more functional requirements of e-voting, including the two mentioned above.

Efficiency of computers is particularly beneficial during the tallying phase, which takes up most of the election time. We could significantly reduce tallying time by employing computers to tally ballots and prepare reports, after which we utilize the network for fast transmission between levels of election officials. But this stage is a critical phase. Preliminary results are vulnerable to attacks when in transmission. Furthermore, during the tallying process, when the results are exposed, authorized entities with malicious intention could tamper with the reports, swinging the outcome to their favored candidate. In our opinion, the electronic ballots are at the most risk when in this stage.

To gain trust, researchers scrutinize deployed e-voting systems and verify their security properties. This is mainly done by reviewing the source code of the voting machines [14, 12, 27, 5, 22]. However, analyzing hundreds and even thousands of lines of code is a daunting task that requires significant amount of resources. Other researchers choose formal verification techniques to analyze e-voting schemes [16, 8, 7, 9, 15, 2, 26]. These schemes are formally specified, after which analysis is done on the logic of the schemes. Working with the formal representation will show the behaviour of the scheme when deployed, while verification in high-level representation will make it easier to discover faults and address them. The challenge is in modelling the schemes

[1]The first reference to the notion of e-voting in computer science literature is in [6]. Although not a complete system or protocol description, in 1981, Chaum mentioned about using his anonymity scheme in e-voting.

precisely in order to verify them accurately.

In this paper, we proceed with a similar framework presented in [16, 8], that is we formally model and analyze e-voting protocols in the applied pi-calculus [1]. Instead of the voting process, we focus on the transmission phase of e-voting. This paper presents the formal specification of two electronic ballot transmission protocols in the applied pi-calculus. We also discuss standard secrecy property of the protocols, which was verified using the tool, ProVerif [3].

The paper is structured as follows. In Section 2, we describe the two schemes that are to be analyzed. Section 3 discusses the formal technique that we choose to utilize. We present the formal models in Section 4. Finally, a discussion on the results of analysis in Section 5.

2. TWO ELECTRONIC BALLOT TRANSMISSION SCHEMES

2.1 Scheme1

Consider the first electronic ballot transmission scheme that we will call Scheme1, given in Figure 1, where $0, \ldots, n$ are election authorities from different governmental levels, DC is the data center to store and publish preliminary results. We generalize election authorities to 4 levels with different tasks described below. Connection refers to Internet connection using any available means, e.g. Cable, DSL, 3G, GPRS, etc.

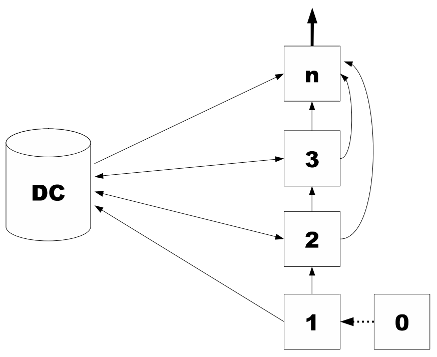

Figure 1: Ballot transmission scheme 1.

2.1.1 *Informal description*

Level 0 Polling place level, where authorities tally the votes manually then describe the results in paper reports. We assume that there is no connection in this level, so reports are transported to the next level using the conventional means, represented by the dotted line in Figure 1.

Level 1 This level is responsible for encoding the reports gathered from Level 0's in its vicinity. We assume that there should be connection starting at this level. After uploading Level 0 data, authorities tally the votes then forward the result to the next level, and upload it to the Data Center.

Level 2 Authorities retrieve Level 1 data from the Data Center and check if they match directly forwarded reports, then tally of Level 1 results is performed. We assume that there are final results starting this level, so the authorities report it to the national level. The remaining results will be forwarded and uploaded.

Level 3 Authorities in this level performs tallying based on reports from the previous level, which have been matched to the information from the Data Center, then they report the final result and the remaining result to the national level.

Level n National level, where authorities receive reports from Level 3's, match them to the Data Center, then perform tally. Finally, the national authority will declare all final results, which includes the Level 2 final result forwarded earlier and the Level 3 final result.

2.1.2 *Security considerations*

We address three basic security properties with standard cryptographic solutions for Scheme1. Confidentiality is addressed by implementing asymmetric encryption. Integrity and Authentication are addressed by employing digital signature. We generalize these cryptographic solutions used by not specifying particular schemes, instead we only define representing functions. Any suitable scheme could be employed on implementation as long as it is an asymmetric encryption scheme that supports digital signatures, such as RSA [23] or ElGamal [11].

In this scheme, we could consider two types of communication: Authority-Authority and Authority-Data Center. Since information in the Data Center is used only for storage and verification, we only perform encryption on the data being communicated. Any anomalies could be discovered when the data is verified by authorities after which it could be corrected accordingly. We handle Authority-Authority communication with more caution, that is we secure it using encryption and digital signature because data received through this communication line is considered the official result.

2.1.3 *The protocol*

Applying the cryptographic protocols and considering the communication types, we get the following ballot transmission protocol. R_i and F_i are respectively preliminary and final result from level i. pk_x and sk_x are respectively public and private key of entity x. $\mathtt{enc}(m, k)$ and $\mathtt{sign}(m, k)$ are respectively functions to encrypt and sign message m using key k.

1. $1 \to DC : \mathtt{enc}(R_0, pk_{DC})$
2. $1 \to 2 : \mathtt{enc}(\mathtt{sign}(R_1, sk_1), pk_2)$
3. $1 \to DC : \mathtt{enc}(R_1, pk_{DC})$
4. $DC \to 2 : \mathtt{enc}(R_1, pk_2)$
5. $2 \to 3 : \mathtt{enc}(\mathtt{sign}(R_2, sk_2), pk_3)$
6. $2 \to DC : \mathtt{enc}(R_2, pk_{DC})$
7. $2 \to n : \mathtt{enc}(\mathtt{sign}(F_2, sk_2), pk_n)$
8. $DC \to 3 : \mathtt{enc}(R_2, pk_3)$
9. $3 \to n : \mathtt{enc}(\mathtt{sign}((R_3, F_3), sk_3), pk_n)$
10. $3 \to DC : \mathtt{enc}(R_3, pk_{DC})$
11. $DC \to n : \mathtt{enc}(R_3, pk_n)$
12. $n \to \ : (F_2, F_3, R_n)$

2.2 Scheme2

The second electronic ballot transmission scheme, which we will call Scheme2, is given in Figure 2, where $1,\ldots,n$ are authorities, CS and AS are the central and auxiliary servers, respectively. CS acts as a data storage that is accessible to the election officials, while AS is accessible to other entities that are involved in the election.

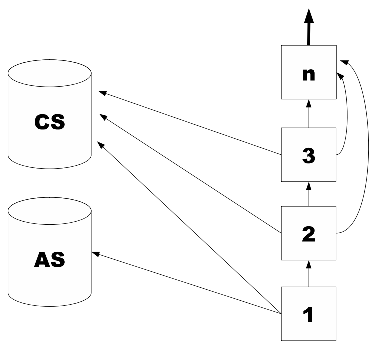

Figure 2: Ballot transmission scheme 2.

2.2.1 *Informal description*

Entities in this scheme proceed in a similar way to Scheme1, except for the following cases.

- Voting machines are deployed in Level 1, which is the polling place level in this scheme, to tally the votes and are capable of transmitting the reports. It forwards the results to the next level and the two servers.
- Level 2 and 3 updates CS only.
- Level 2 to n does not retrieve data from any server.

2.2.2 *Security considerations*

Scheme2 emphasizes on Integrity and Authentication by employing one-way hash function and digital signature. Any hash function could be employed as long as it is compatible for use with digital signature, such as SHA [21].

Similar to the previous scheme, there are two different types of communication in this scheme: Authority-Authority and Authority-Server. Authority-Authority communication that conveys the official result is secured using hash function and digital signature. For experimental purposes, we use encryption only on the Authority-Server communication. In implementation, both types should be encrypted.

2.2.3 *The protocol*

The protocol for Scheme2 is given below. R_i and F_i are respectively preliminary and final result from level i. pk_x and sk_x are respectively public and private key of entity x. $\mathtt{enc}(m,k)$ and $\mathtt{sign}(k,m)$ are respectively functions to encrypt and sign message m using key k. $\mathtt{h}(m)$ is the one-way hash function applied to message m.

1. $1 \rightarrow 2 : (R_1, \mathtt{sign}(\mathtt{h}(R_1), sk_1))$
2. $1 \rightarrow CS : \mathtt{enc}(R_1, pk_{CS})$
3. $1 \rightarrow AS : \mathtt{enc}(R_1, pk_{AS})$
4. $2 \rightarrow n : (F_2, \mathtt{sign}(\mathtt{h}(F_2), sk_2))$
5. $2 \rightarrow 3 : (R_2, \mathtt{sign}(\mathtt{h}(R_2), sk_2))$
6. $2 \rightarrow CS : \mathtt{enc}(R_2, pk_{CS})$
7. $3 \rightarrow n : ((F_3, R_3), \mathtt{sign}(\mathtt{h}((F_3, R_3)), sk_3))$
8. $3 \rightarrow CS : \mathtt{enc}(R_3, pk_{CS})$
9. $n \rightarrow \quad : (F_2, F_3, R_n)$

3. APPLIED PI-CALCULUS

The applied pi-calculus [1] is a language to model communication and concurrency of processes. It builds on pi-calculus [20] but introduce new semantics to facilitate the study of security protocols. The calculus allows messages to be constructed from names and functions, which gives us means to model cryptographic protocols appropriately. This section provides a basic review of the calculus' syntax and operational semantics.

3.1 Syntax

In applied pi-calculus, we define terms L, M, N, T, U, V from a set of names $a, b, c, \ldots, k, \ldots, m, n, \ldots, s$, a set of variables x, y, z, and function symbols in a *signature* Σ. Function symbols are of form $f(M_1, \ldots, M_l)$, where l corresponds to the arity of function f. When considering cryptographic primitives, we can have the function symbol $\mathtt{enc}$ to model the process of encryption, which returns an encrypted message with the message and a key as its parameters. We could also have a corresponding $\mathtt{dec}$ function that returns the message, given an encrypted message and the key. The signature Σ has an equational theory E, that defines the equality of terms denoted by $=_E$. If we model symmetric encryption with the two functions mentioned above, we can have the equational theory $\mathtt{dec}(\mathtt{enc}(x,k),k) = x$, then we can say that $\mathtt{dec}(\mathtt{enc}(\mathtt{dec}(\mathtt{enc}(m,k_2),k_2),k_1),k_1) =_E m$.

The grammar of processes are given in Table 1. Plain processes are similar to processes of pi-calculus. Extended processes are the novelty of applied pi-calculus.

The calculus includes *active substitution* $\{M/x\}$ that replaces variable x with term M. To control its scope, we write $\nu x.(\{M/x\} \mid P)$ that corresponds exactly to let $x = M$ in P. Names and variables have scopes. We write $fn(A)$ and $bn(A)$ for the set of free and bound names of A, $fv(A)$ and $bv(A)$ for the set of free and bound variables of A.

The notation in$(u,\ =M)$ is used to check if the input on channel u is equal to term M. The equality considered is with respect to E.

Given an extended process A, if we replace all plain process with 0, we obtain the frame of A, $\varphi(A)$. A frame is an extended process built from 0 and active substitutions only by parallel composition and restriction. The frame of A represent the static knowledge A exposes to the environment. The domain of a frame, $dom(\varphi)$, is the set of variables that the frame exports through substitution.

A context, written $C[\_]$, is defined as an extended process with a hole. An *evaluation context* is a context whose hole

Table 1: Applied pi-calculus grammar.

$P, Q, R ::=$	plain processes	$A, B, C ::=$	extended processes
0	null process	P	plain process
$P \mid Q$	parallel composition	$A \mid B$	parallel composition
$!P$	replication	$\nu n.A$	name restriction
$\nu n.P$	name restriction	$\nu x.A$	variable restriction
$u(x).P$	message input	$\{M/x\}$	active substitution
$\overline{u}\langle M\rangle.P$	message output		
if $M = N$ then P else Q	conditional		

is not under a replication, a conditional, an input, or an output.

3.2 Semantics

The operational semantics of applied pi-calculus consist of the three relations that are described below.

Structural equivalence $\equiv$ is the smallest equivalence relation on extended processes that is closed by α-conversion on both names and variables and application of evaluation contexts such that:

Par-0	A	$\equiv$	$A \mid 0$
Par-A	$A \mid (B \mid C)$	$\equiv$	$(A \mid B) \mid C$
Par-C	$A \mid B$	$\equiv$	$B \mid A$
Repl	$!P$	$\equiv$	$P \mid !P$
New-0	$\nu n.0$	$\equiv$	0
New-C	$\nu u.\nu v.A$	$\equiv$	$\nu v.\nu u.A$
New-Par	$A \mid \nu u.B$	$\equiv$	$\nu u.(A \mid B)$ if $u \notin fv(A) \cup fn(A)$
Alias	$\nu x.\{M/x\}$	$\equiv$	0
Subst	$\{M/x\} \mid A$	$\equiv$	$\{M/x\} \mid A\{M/x\}$
Rewrite	$\{M/x\}$	$\equiv$	$\{N/x\}$ if $M =_E N$

Internal reduction $\rightarrow$ is the smallest relation closed by structural equivalence and application of evaluation contexts such that:

Comm	$\bar{a}\langle x\rangle.P \mid a(x).Q$	$\rightarrow$	$P \mid Q$
Then	if $M = N$ then P else Q	$\rightarrow$	P
Else	if $M = N$ then P else Q	$\rightarrow$	Q

for ground terms M, N where $M \neq_E N$

Labelled reduction $\xrightarrow{\alpha}$ extends internal reduction, allowing processes to interact with the environment using the following rules:

In	$a(x).P \xrightarrow{a(M)} P\{M/x\}$
Out-Atom	$\bar{a}\langle u\rangle.P \xrightarrow{\bar{a}\langle u\rangle} P$
Open-Atom	$\dfrac{A \xrightarrow{\bar{a}\langle u\rangle} A' \qquad u \neq a}{\nu u.A \xrightarrow{\nu u.\bar{a}\langle u\rangle} A'}$
Scope	$\dfrac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$
Par	$\dfrac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$
Struct	$\dfrac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'}$

3.3 Equivalences

In analysis of security protocols, some properties are formally proven by showing equivalence of processes. We often use the notion of *observational equivalence*, that is processes cannot be distinguished by any context. We briefly describe several equivalences in applied pi-calculus.

Static equivalence $\approx_s$ relates frames that cannot be distinguished, and processes with respect to the static knowledge they expose to the environment.

Labelled bisimilarity $\approx_l$ relates extended processes that cannot be distinguished in terms of labelled transitions.

It has been shown that labelled bisimilarity and observational equivalence in applied pi-calculus coincides. We refer interested readers to [1] for more deliberation on this relation. Readers will also find the formal definitions of the equivalences mentioned above.

4. MODELLING TRANSMISSION PROTOCOLS IN APPLIED PI-CALCULUS

In this section, we present the transmission schemes in Section 2 modelled in the applied pi-calculus. We annotate the correspondence between the protocols[2] and the models with comments[3]. A comment is placed before the output process corresponding to the step in the protocol.

[2]Sections 2.1.3 and 2.2.3
[3]presented in (* *)

4.1 Scheme1

4.1.1 Signature and equational theory

We start modelling the protocol of Scheme1 by defining its signature $\Sigma = \{\mathtt{pk}, \mathtt{enc}, \mathtt{dec}, \mathtt{sign}, \mathtt{checksign}\}$. These functions build the equational theory E:

$$\mathtt{dec}(\mathtt{enc}(x, \mathtt{pk}(y)), y) = x$$
$$\mathtt{checksign}(\mathtt{pk}(y), \mathtt{sign}(x, y)) = x$$

We model asymmetric encryption with two functions, a constructor `enc` and the corresponding destructor `dec`. To encrypt, let the message x and the public key of the destination entity, produced by performing the one-way function `pk` on secret key y, be parameters to function `enc`. The output, namely the encrypted message, is decrypted using function `dec`, also 2-tuple, with the secret key as the second argument. We proceed in a similar way for digital signature, modelled by functions `sign` and `checksign` for, respectively, signing and extracting a message.

4.1.2 Main process

The main process declares the private channels, mostly for key distribution, and all other processes in parallel. We model the numerous Level 1, 2, and 3's by replicating their processes.

$$\begin{aligned} P \triangleq\ & \nu pkCh1.\nu pkCh2.\nu pkCh3.\nu pkChn.\nu pkChDC. \\ & \nu skCh1.\nu skCh2.\nu skCh3.\nu skChn.\nu skChDC. \\ & (K \mid !A_1 \mid !A_2 \mid !A_3 \mid A_n \mid DC) \end{aligned}$$

4.1.3 Key Administration process

We follow the notion in [9] to include a key distribution process that acts like a PKI[4]. First, we create fresh private keys, then apply the `pk` function to get the public keys. Next, all keys are distributed through private channels so an attacker cannot intercept or inject bogus keys. Our assumption is in an election setting, only authorities know about the keys, so we modelled this by using private channels[5].

$$\begin{aligned} K \triangleq\ & \nu sk_1.\nu sk_2.\nu sk_3.\nu sk_n.\nu sk_{DC}. \\ & \{\mathtt{pk}(sk_1)/pk_1\} \mid \overline{pkCh1}\langle pk_1\rangle \mid \\ & \{\mathtt{pk}(sk_2)/pk_2\} \mid \overline{pkCh2}\langle pk_2\rangle \mid \\ & \{\mathtt{pk}(sk_3)/pk_3\} \mid \overline{pkCh3}\langle pk_3\rangle \mid \\ & \{\mathtt{pk}(sk_n)/pk_n\} \mid \overline{pkChn}\langle pk_n\rangle \mid \\ & \{\mathtt{pk}(sk_{DC})/pk_{DC}\} \mid \overline{pkChDC}\langle pk_{DC}\rangle \mid \\ & \overline{skCh1}\langle sk_1\rangle \mid \overline{skCh2}\langle sk_2\rangle \mid \overline{skCh3}\langle sk_3\rangle \mid \\ & \overline{skChn}\langle sk_n\rangle \mid \overline{skChDC}\langle sk_{DC}\rangle \end{aligned}$$

4.1.4 Authority processes

We define a dedicated process for each level authority. The processes starts with obtaining required keys from the PKI. We model results from Level 0's by declaring a new variable R_0 inside process A_1. Intuitively, this represents Result 0 freshly encoded (electronically created) by Authority 1. Other procedures in the processes are receiving results from previous levels, decryption and signature verification, matching received result to the server data, and creating new variables representing the level's report. Finally, the reports are sent to the required entities. Notice that for signature verification, along with the signature, we send the public key of the source *src*, $\mathtt{pk}(sk_{src})$, to prove their identity. The destination entity will first check if the sent public key match the one published by the PKI. To end the protocol, the national authority, A_n, publishes the results, which represents declaring the official outcome of the election.

[4]Public Key Infrastructure

[5]In some of our analysis, we assume public channels for distributing public keys, most notably in Section 5.3

$$\begin{aligned} A_1 \triangleq\ & skCh1(sk_1). \\ & pkCh2(pk_2).pkChDC(pk_{DC}). \\ & \nu R_0. \\ & \texttt{(* Step 1 *)} \\ & \overline{c}\langle \mathtt{enc}(R_0, pk_{DC})\rangle. \\ & \nu result1. \\ & \texttt{(* Step 2 *)} \\ & \overline{c}\langle \mathtt{enc}((\mathtt{pk}(sk_1), \mathtt{sign}(result1, sk_1)), pk_2)\rangle. \\ & \texttt{(* Step 3 *)} \\ & \overline{c}\langle \mathtt{enc}(result1, pk_{DC})\rangle \end{aligned}$$

$$\begin{aligned} A_2 \triangleq\ & skCh2(sk_2).pkCh1(pk_1). \\ & pkCh3(pk_3).pkChn(pk_n).pkChDC(pk_{DC}). \\ & !c(m1).!c(t1). \\ & \{\mathtt{dec}(m_1, sk_2)/(pubkey1, signed1)\} \mid \\ & \text{if } pubkey1 = pk_1 \text{ then} \\ & \{\mathtt{checksign}(signed1, pubkey1)/R_1\} \mid \\ & \{\mathtt{dec}(t1, sk_2)/D1\} \mid \\ & \text{if } R_1 = D1 \text{ then} \\ & \nu result2.\nu final2. \\ & \texttt{(* Step 5 *)} \\ & \overline{c}\langle \mathtt{enc}((\mathtt{pk}(sk_2), \mathtt{sign}(result2, sk_2)), pk_3)\rangle. \\ & \texttt{(* Step 6 *)} \\ & \overline{c}\langle \mathtt{enc}(result2, pk_{DC})\rangle. \\ & \texttt{(* Step 7 *)} \\ & \overline{c}\langle \mathtt{enc}((\mathtt{pk}(sk_2), \mathtt{sign}(final2, sk_2)), pk_n)\rangle \end{aligned}$$

$$\begin{aligned} A_3 \triangleq\ & skCh3(sk_3). \\ & pkCh2(pk_2).pkChn(pk_n).pkChDC(pk_{DC}). \\ & !c(m2).!c(t2). \\ & \{\mathtt{dec}(m2, sk_3)/(pubkey2, signed2)\} \mid \\ & \text{if } pubkey2 = pk_2 \text{ then} \\ & \{\mathtt{checksign}(signed2, pubkey2)/R_2\} \mid \\ & \{\mathtt{dec}(t2, sk_3)/D2\} \mid \\ & \text{if } R_2 = D2 \text{ then} \\ & \nu result3.\nu final3. \\ & \texttt{(* Step 9 *)} \\ & \overline{c}\langle \mathtt{enc}((\mathtt{pk}(sk_3), \\ & \quad \mathtt{sign}((result3, final3), pk_3)), pk_n)\rangle. \end{aligned}$$

$$
\begin{aligned}
&\texttt{(* Step 10 *)}\\
&\bar{c}\langle \mathtt{enc}(result3, pk_{DC})\rangle
\end{aligned}
$$

$$
\begin{aligned}
A_n \;\triangleq\;& skChn(sk_n).\\
& pkCh2(pk_2).pkCh3(pk_3).\\
& !c(m2).!c(m3).!c(t3).\\
& \{\mathtt{dec}(m2, sk_n)/(pubkey2, signed2)\} \mid\\
& \text{if } pubkey2 = pk_2 \text{ then}\\
& \{\mathtt{checksign}(signed2, pubkey2)/F_2\} \mid\\
& \{\mathtt{dec}(m3, sk_n)/(pubkey3, signed3)\} \mid\\
& \text{if } pubkey3 = pk_3 \text{ then}\\
& \{\mathtt{checksign}(signed3, pubkey3)/(R_3, F_3)\} \mid\\
& \{\mathtt{dec}(t3, sk_n)/D3\} \mid\\
& \text{if } R_3 = D3 \text{ then}\\
& \nu R_n.\\
& \texttt{(* Step 12 *)}\\
& \bar{c}\langle (F_2, F_3, R_n)\rangle
\end{aligned}
$$

4.1.5 *Data Center process*

This process also starts by obtaining keys from the PKI, then accepts results for storage and sends stored results for verification. Result 0 does not need to be sent out, since Authority 1 have it and can perform the required processes on it. We consider the raw Result 0's, handled by Authority 1, as the most trusted data and the stored form to be the most trusted electronic reference.

$$
\begin{aligned}
DC \;\triangleq\;& skChDC(sk_{DC}).\\
& pkCh2(pk_2).pkCh3(pk_3).pkChn(pk_n).\\
& c(m0).c(m1).\\
& \texttt{(* Step 4 *)}\\
& \bar{c}\langle \mathtt{enc}(\mathtt{dec}(m1, sk_{DC}), pk_2)\rangle.\\
& c(m2).\\
& \texttt{(* Step 8 *)}\\
& \bar{c}\langle \mathtt{enc}(\mathtt{dec}(m2, sk_{DC}), pk_3)\rangle.\\
& c(m3).\\
& \texttt{(* Step 11 *)}\\
& \bar{c}\langle \mathtt{enc}(\mathtt{dec}(m3, sk_{DC}), pk_n)\rangle
\end{aligned}
$$

4.2 Scheme2

For brevity, we do not describe the main process and the key administration process on this model of Scheme2. These processes are similar to those found in Sections 4.1.2 and 4.1.3.

4.2.1 *Signature and equational theory*

The model of Scheme2 in applied pi-calculus has the signature $\Sigma = \{\mathtt{h}, \mathtt{pk}, \mathtt{enc}, \mathtt{dec}, \mathtt{sign}, \mathtt{checksign}\}$. These functions build the equational theory E:

$$
\begin{aligned}
\mathtt{dec}(\mathtt{enc}(x, \mathtt{pk}(y)), y) &= x\\
\mathtt{checksign}(\mathtt{pk}(y), \mathtt{sign}(\mathtt{h}(x), y)) &= \mathtt{h}(x)
\end{aligned}
$$

The functions work in the same way as those in Section 4.1.1, except we can extract a hash of the message x from the digital signature.

4.2.2 *Authority processes*

The processes starts with obtaining required keys from the PKI. The next processes are receiving results from previous levels, signature and hash verification, then the authority creates a new variable representing the level's report. Finally, the reports are sent to the required entities.

$$
\begin{aligned}
A_1 \;\triangleq\;& skCh1(sk_1).\\
& pkChCS(pk_{CS}).pkChAs(pk_{AS}).\\
& \nu result1.\\
& \texttt{(* Step 1 *)}\\
& \bar{c}\langle (result1, (\mathtt{pk}(sk_1), \mathtt{sign}(\mathtt{h}(result1), sk_1)))\rangle.\\
& \texttt{(* Step 2 *)}\\
& \bar{c}\langle \mathtt{enc}(result1, pk_{CS})\rangle.\\
& \texttt{(* Step 3 *)}\\
& \bar{c}\langle \mathtt{enc}(result1, pk_{AS})\rangle
\end{aligned}
$$

$$
\begin{aligned}
A_2 \;\triangleq\;& skCh2(sk_2).\\
& pkCh1(pk_1).pkChDC(pk_{CS}).\\
& c(m1).\{(R_1, sign1)/m1\} \mid\\
& \{(pubkey1, signed1)/sign1\} \mid\\
& \text{if } pubkey1 = pk_1 \text{ then}\\
& \{hash1/\mathtt{checksign}(signed1, pubkey1)\} \mid\\
& \text{if } hash1 = \mathtt{h}(R_1) \text{ then}\\
& \nu final2.\nu result2.\\
& \texttt{(* Step 4 *)}\\
& \bar{c}\langle (final2, (\mathtt{pk}(sk_2), \mathtt{sign}(\mathtt{h}(final2), sk_2)))\rangle.\\
& \texttt{(* Step 5 *)}\\
& \bar{c}\langle (result2, (\mathtt{pk}(sk_2), \mathtt{sign}(\mathtt{h}(result2), sk_2)))\rangle.\\
& \texttt{(* Step 6 *)}\\
& \bar{c}\langle \mathtt{enc}(result2, pk_{CS})\rangle
\end{aligned}
$$

$$
\begin{aligned}
A_3 \;\triangleq\;& skCh3(sk_3).\\
& pkCh1(pk_3).pkChDC(pk_{CS}).\\
& c(m2).\{(R_2, sign2)/m2\} \mid\\
& \{(pubkey2, signed2)/sign2\} \mid\\
& \text{if } pubkey2 = pk_2 \text{ then}\\
& \{hash2/\mathtt{checksign}(signed2, pubkey2)\} \mid\\
& \text{if } hash2 = \mathtt{h}(R_2) \text{ then}\\
& \nu final3.\nu result3.\\
& \texttt{(* Step 7 *)}\\
& \bar{c}\langle ((final3, result3),\\
& \quad (\mathtt{pk}(sk_3), \mathtt{sign}(h(final3, result3), sk_3)))\rangle.\\
& \texttt{(* Step 8 *)}\\
& \bar{c}\langle \mathtt{enc}(result3, pk_{CS})\rangle
\end{aligned}
$$

$$\begin{aligned}
A_n \;\triangleq\;& skChn(sk_n).\\
& pkCh2(pk_2).pkCh3(pk_3).\\
& c(m2).\{(F_2, sign2)/m2\} \mid\\
& \{(pubkey2, signed2)/sign2\} \mid\\
& \text{if } pubkey2 = pk_2 \text{ then}\\
& \{hash2/\texttt{checksign}(signed2, pubkey2)\} \mid\\
& \text{if } hash2 = \texttt{h}(F_2) \text{ then}\\
& c(m3).\{(FR3, sign3)/m3\} \mid\\
& \{(pubkey3, signed3)/sign3\} \mid\\
& \text{if } pubkey3 = pk_3 \text{ then}\\
& \{hash2/\texttt{checksign}(signed3, pubkey3)\} \mid\\
& \text{if } hash2 = \texttt{h}(FR3) \text{ then}\\
& \{(F_3, R_3)/FR3\} \mid\\
& \nu R_n.\\
& \texttt{(* Step 9 *)}\\
& \overline{c}\langle((F_2, F_3, R_n))\rangle
\end{aligned}$$

4.2.3 *Server processes*
These processes only receives results from authorities, then stores it. CS receives results from A_1, A_2, and A_3, while AS from A_1 only.

$$CS \triangleq c(m1).c(m2).c(m3)$$

$$AS \triangleq c(m1)$$

5. ANALYSIS AND DISCUSSION
In this section we present our analysis of the ballot transmission protocols given above in the applied pi-calculus. Most of the properties are verified using ProVerif. Most of our claims below are based on ProVerif verification outcomes.

5.1 ProVerif
When employing applied pi-calculus to verify security properties, we can use techniques given in [1] to perform hand proving, and/or we can utilize the tool ProVerif by Bruno Blanchet [3] to derive automated proofs. An advantage of ProVerif that is useful to this research is that it is not restricted to a bounded number of sessions. We utilize this in modelling the numerous regions per administrative level in the election system, which is done by replicating processes corresponding to the level. Like other model checkers, ProVerif can perform *reachability* test that we can use to verify secrecy properties. It can also be used to verify authentication properties using *correspondence* test and some test based on *observational equivalence*, which are used to verify privacy properties.

To make the input scripts for ProVerif, the schemes' applied pi-calculus models in Section 4 are slightly modified. The modifications are relatively straight forward. Each applied pi-calculus command has its equivalent in ProVerif input script style. A complete listing of all commands in ProVerif and its usage can be found in the user manual [4].

Attacker. In the calculus' setting, only honest entities are modelled. The attacker is considered to be a part of the environment. In ProVerif, the attacker is in control of the network limited by perfect cryptography, following the Dolev-Yao model [10]. So, given a public communication channel, an attacker can eavesdrop and intercept (passive attacker) even modify and send messages (active attacker) through it. In our case, we have the public channel c where attacks could be performed.

Assumptions
In this paper, our assumption is that the transmission itself is perfect. That is the equipments, communication lines, and other factors that might affect the transmission are working as they are supposed to.

In the model, the data being handled are formally specified in a high-level representation, we do not address it in its bit representation. We assume that the equipments used are capable of low-level error correction. The model enforces certain formats for the messages being sent. Any ill-formed messages will safely be ignored. We also present a technique to check the data being sent (verification of data). If we keep a trusted copy of the data, then we can perform verification by comparing the transmitted data with the stored copy. This could be seen in the model of Scheme1.

5.2 Standard secrecy
We check standard secrecy based on reachability, which is a basic feature of ProVerif. That is, we test whether a free variable that represents a certain secret information cannot be deduced by the attacker. We request ProVerif to verify the secrecy of all keys and results (preliminary and final).

Confidentiality of keys. First, we request ProVerif to check standard secrecy for all keys used in the protocols. Since the keys are distributed through a private channel, we expect them to be secure. ProVerif showed that all keys cannot be derived by an attacker.

Confidentiality of results
Next, we query for the standard secrecy of all results communicated and processed in the protocols. For Scheme1, the output showed that all ballot transmissions are secure, while for Scheme2, the transmitted results are derivable, in other words they are exposed to an attacker. This is due to the absence of encryption in the latter, which we deliberately left as an unsecured transmission. This also shows what could happen if security features are not implemented. We could further modify the model to not use digital signature, which would trivially threaten the secrecy of the results transmitted even more.

The final results published by Authority n were also derivable. We must perform a different technique to check its secrecy, which could be done by segregating the model in phases using the separation command `phase`. This is part of the next task in the study.

While verifying for secrecy, we came upon the case of corrupt administrator. Further discussion on this result is given below.

5.3 Corrupt administrator

We consider the threat of corrupt authorities to be critical in the protocol. In [16] and [8], corrupt administrators were modelled by outputting their secret keys, so attackers can perform actions that should have been administrator privileges. This subsection describes our analysis of the protocol with respect to the case of corrupt administrator.

On R_0 of Scheme1. Firstly, we assume that R_0 was safely transported to Authority 1. Since Level 0 results are in paper form, there does not exist an electronic attack towards them. Concerns should include physical, social engineering, and other attacks which are well beyond the scope of this paper. The results are at most risk when the R_0's are with Authority 1. R_0 will be the first electronic data that will be upload, thus if any information is modified, then indeed the next results will be skewed. We assume that at Level 0, party representatives and other witnesses observed the manual tally, signed the reports, and kept a hardcopy of the results for verification. When Authority 1 uploads the R_0's, a participating party, for instance, could verify the published result with their own recapitulation, consolidating records from their Level 0 representatives. Any discrepancies could be identified easily and reported for corrections.

Each R_i in Scheme1, where $0 < i < n$, is at risk when Authorities $i + 1$ decrypt and process them. In our analysis, when checking the secrecy of results handled by the administrators, we observe that there exist a possible attack on the protocol. The attack is related to the corrupt administrator case. In reality, this case happens when a dishonest authority leaks his secret key to an unauthorized person with malicious intentions. The attacker can easily retrieve the results by decrypting using the authority's secret key and gain knowledge of the results before it is officially published. In the model, we simulate this case by outputting the authority's secret key.

Note that for Scheme2, it is not necessary to formally show the existence of the corrupt administrator case. Since the results are derivable and no encryption is employed, an attacker does not need a key to decrypt the result transmitted. So, the case of corrupt administrator is not applicable to Scheme2.

As a case study, based on process A_2 from Section 4.1.4, we show an attacker process wherein R_1 is derivable.

$$\begin{aligned} A_{2x} \triangleq\ & chx(a1).c(pk_1).c(m1).\\ & \{\mathtt{dec}(m1,a1)/(pubkey1, signed1)\} \mid \\ & \text{if } pubkey1 = pk_1 \text{ then}\\ & \{\mathtt{checksign}(signed1, pubkey1)/R_1\} \mid \overline{c}\langle R_1\rangle \end{aligned}$$

The attacker has `a_1`, which is a handler for sk_2, received through chx, a channel between the dishonest authority and the attacker. The public key of Authority 1 can be retrieved publicly through channel[6] c. He also retrieves $m1$ from the public channel, then proceeds to derive R_1. Finally, he publishes R_1 to signify his success.

[6]The main model uses private channels to distribute public keys. For this instance, we could modify the model to use public channels instead. Note that distributing the public keys through public channel will not affect the security of the protocol.

We present this scenario formally by disproving *Reachability-based secrecy* as defined in [25]. Consider process A_1 of Scheme1 and the attacker process A_{2x} above. Let Q be some process. We prove that context $C[\overline{c}\langle R_1\rangle.Q]$ is derivable. For brevity, we perform several steps in some of the displayed transitions, all using the operational semantics of applied pi-calculus.

$$\begin{aligned}
& A_1 \mid A_{2x}\\
\equiv\ & C[skCh1(sk_1).\\
& \quad pkCh2(pk_2).pkChDC(pk_{DC}).\\
& \quad \nu R_0.\overline{c}\langle \mathtt{enc}(R_0, pk_{DC})\rangle.\\
& \quad \nu result1.\\
& \quad \overline{c}\langle \mathtt{enc}((\mathtt{pk}(sk_1), \mathtt{sign}(result1, sk_1)), pk_2)\rangle.\\
& \quad \overline{c}\langle \mathtt{enc}(result1, pk_{DC})\rangle\\
& \quad \mid\ chx(a1).c(pk_1).c(m1).\\
& \qquad \{\mathtt{dec}(m1,a1)/(pubkey1, signed1)\} \mid\\
& \qquad \text{if } pubkey1 = pk_1 \text{ then}\\
& \qquad \{\mathtt{checksign}(signed1, pubkey1)/R_1\} \mid\\
& \qquad \overline{c}\langle R_1\rangle]\\
\xrightarrow{*}\ & C[\nu R_0.\overline{c}\langle \mathtt{enc}(R_0, pk_{DC})\rangle.\\
& \quad \nu result1.\\
& \quad \overline{c}\langle \mathtt{enc}((\mathtt{pk}(sk_1), \mathtt{sign}(result1, sk_1)), pk_2)\rangle.\\
& \quad \overline{c}\langle \mathtt{enc}(result1, pk_{DC})\rangle\\
& \quad \mid\ chx(a1).c(pk_1).c(m1).\\
& \qquad \{\mathtt{dec}(m1,a1)/(pubkey1, signed1)\} \mid\\
& \qquad \text{if } pubkey1 = pk_1 \text{ then}\\
& \qquad \{\mathtt{checksign}(signed1, pubkey1)/R_1\} \mid\\
& \qquad \overline{c}\langle R_1\rangle]\\
\equiv\ & C[\nu R_0, x_1.(\overline{c}\langle x_1\rangle \mid \{\ \mathtt{enc}(R_0, pk_{DC})/x_1\}).\\
& \quad \nu result1.\\
& \quad \overline{c}\langle \mathtt{enc}((\mathtt{pk}(sk_1), \mathtt{sign}(result1, sk_1)), pk_2)\rangle.\\
& \quad \overline{c}\langle \mathtt{enc}(result1, pk_{DC})\rangle\\
& \quad \mid\ chx(a1).c(pk_1).c(m1).\\
& \qquad \{\mathtt{dec}(m1,a1)/(pubkey1, signed1)\} \mid\\
& \qquad \text{if } pubkey1 = pk_1 \text{ then}\\
& \qquad \{\mathtt{checksign}(signed1, pubkey1)/R_1\} \mid\\
& \qquad \overline{c}\langle R_1\rangle]\\
\rightarrow\ & C[\nu R_0.\{\mathtt{enc}(R_0, pk_{DC})/x_1\} \mid\\
& \quad \nu result1.\\
& \quad \overline{c}\langle \mathtt{enc}((\mathtt{pk}(sk_1), \mathtt{sign}(result1, sk_1)), pk_2)\rangle.\\
& \quad \overline{c}\langle \mathtt{enc}(result1, pk_{DC})\rangle\\
& \quad \mid\ chx(a1).c(pk_1).c(m1).\\
& \qquad \{\mathtt{dec}(m1,a1)/(pubkey1, signed1)\} \mid\\
& \qquad \text{if } pubkey1 = pk_1 \text{ then}\\
& \qquad \{\mathtt{checksign}(signed1, pubkey1)/R_1\} \mid\\
& \qquad \overline{c}\langle R_1\rangle]\\
\equiv\ & \nu R_0.C[\{\mathtt{enc}(R_0, pk_{DC})/x_1\}.
\end{aligned}$$

$$
\begin{array}{ll}
 & \nu result1, x_2, x_3.((\overline{c}\langle x_2\rangle \mid \\
 & \quad \{\mathtt{enc}((\mathtt{pk}(sk_1), \mathtt{sign}(result1, sk_1)), pk_2)/x_2\}). \\
 & \quad (\overline{c}\langle x_3\rangle \mid \{\mathtt{enc}(result1, pk_{DC})/x_3\})) \\
 & \quad \mid \ \ chx(a1).c(pk_1).c(m1). \\
 & \qquad \{\mathtt{dec}(m1, a1)/(pubkey1, signed1)\} \mid \\
 & \qquad \text{if } pubkey1 = pk_1 \text{ then} \\
 & \qquad \{\mathtt{checksign}(signed1, pubkey1)/R_1\} \mid \\
 & \qquad \overline{c}\langle R_1\rangle] \\
\overset{*}{\rightarrow} & \nu R_0.C[\{\mathtt{enc}(R_0, pk_{DC})/x_1\}.\nu result1. \\
 & \quad (\{\mathtt{enc}((\mathtt{pk}(sk_1), \mathtt{sign}(result1, sk_1)), pk_2)/x_2\}| \\
 & \quad \{\mathtt{enc}(result1, pk_{DC})/x_3\}) \\
 & \quad \mid \ \ chx(a1).c(pk_1).c(m1). \\
 & \qquad \{\mathtt{dec}(m1, a1)/(pubkey1, signed1)\} \mid \\
 & \qquad \text{if } pubkey1 = pk_1 \text{ then} \\
 & \qquad \{\mathtt{checksign}(signed1, pubkey1)/R_1\} \mid \\
 & \qquad \overline{c}\langle R_1\rangle] \\
\equiv & \nu R_0, result1.C[\{\mathtt{enc}(R_0, pk_{DC})/x_1\}| \\
 & \quad \{\mathtt{enc}((\mathtt{pk}(sk_1), \mathtt{sign}(result1, sk_1)), pk_2)/x_2\}| \\
 & \quad \{\mathtt{enc}(result1, pk_{DC})/x_3\} \\
 & \quad \mid \ \ chx(a1).c(pk_1).c(m1). \\
 & \qquad \{\mathtt{dec}(m1, a1)/(pubkey1, signed1)\} \mid \\
 & \qquad \text{if } pubkey1 = pk_1 \text{ then} \\
 & \qquad \{\mathtt{checksign}(signed1, pubkey1)/R_1\} \mid \\
 & \qquad \overline{c}\langle R_1\rangle] \\
\overset{*}{\rightarrow} & \nu R_0, result1.C[\{\mathtt{enc}(R_0, pk_{DC})/x_1\}| \\
 & \quad \{\mathtt{enc}((\mathtt{pk}(sk_1), \mathtt{sign}(result1, sk_1)), pk_2)/x_2\}| \\
 & \quad \{\mathtt{enc}(result1, pk_{DC})/x_3\} \\
 & \quad \mid \ \ \{\mathtt{dec}(m1, a1)/(pubkey1, signed1)\} \mid \\
 & \qquad \text{if } pubkey1 = pk_1 \text{ then} \\
 & \qquad \{\mathtt{checksign}(signed1, pubkey1)/R_1\} \mid \\
 & \qquad \overline{c}\langle R_1\rangle] \\
\equiv & \nu R_0, result1.C[\{\mathtt{enc}(R_0, pk_{DC})/x_1\}| \\
 & \quad \{\mathtt{enc}((\mathtt{pk}(sk_1), \mathtt{sign}(result1, sk_1)), pk_2)/x_2\}| \\
 & \quad \{\mathtt{enc}(result1, pk_{DC})/x_3\} \\
 & \quad \mid \ \ \{\mathtt{dec}(m1, a1)/(pubkey1, signed1)\} \mid \\
 & \qquad \{\mathtt{checksign}(signed1, pubkey1)/R_1\} \mid \\
 & \qquad \overline{c}\langle R_1\rangle]
\end{array}
$$

An attack scenario like the one given above is highly probable to happen in a real implementation. After exposing it in Scheme1, claimed to respect secrecy, we ask *does there exist a transmission protocol that is not vulnerable in the presence of corrupt officials? Does there exist means to mitigate this threat for protocols proven to be susceptible to it?* We intend to address these questions and tackle other issues as well as our work progresses.

Besides corrupt administrators, another case that we could consider is incompetent administrators. That is officials handling some e-voting processes that does not have enough information on how to perform them. As stated in our assumptions, we consider the transmissions to be perfect, and this includes the capability of the entities handling the processes. However, if we receive reports of such scenarios, then it could be part of our future work to formalize its detailed description.

6. CONCLUSION

The transmission phase of e-voting is a critical part of the system. Preliminary results are at risk of alteration during this stage, which could severely impact the outcome of the election. In this paper, we presented the formal specification of two ballot transmission protocols in the applied pi calculus. We then verified the standard secrecy property by reachability tests using ProVerif. The automated proofs showed that the secret keys used in the protocols are secure and when properly encrypted the transmission of the results are safe. While performing secrecy verification, we came upon the case of corrupt administrator. We proceeded to formally define such scenario, and show its possible existence in Scheme1. In our future work, we intend to address this issue. We also plan to verify other secrecy properties and investigate privacy properties (i.e. properties analyzed using *equivalences*). The formal specifications presented here will be further refined in our next analysis, and could serve as a framework for other protocols to be studied. After analyzing the case studies presented in this paper, this framework should be capable to perform analysis of a real case, such as the transmission protocol that have been implemented in a national election.

7. ACKNOWLEDGEMENTS

We would like to thank Pablo Manalastas and Felix Muga for an interesting and helpful discussion about election procedures and its challenges.

8. REFERENCES

[1] M. Abadi and C. Fournet. Mobile Values, New Names, and Secure Communication. In *Proceedings of the 28th Annual ACM SIGPLAN - SIGACT Symposium on Principles of Programming Languages (POPL '01)*, pages 104–115, London, UK, Jan. 2001.

[2] M. Backes, C. Hriţcu, and M. Maffei. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-Calculus. In *Proceedings of 21st IEEE Computer Security Foundations Symposium (CSF '08)*, pages 195–209, Pittsburgh, PA, USA, June 2008.

[3] B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW '01)*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 2001.

[4] B. Blanchet. *ProVerif Automatic Cryptographic Protocol Verifier User Manual*, 14 Sept. 2010.

[5] California Secretary of State Debra Bowen. "Top-to-Bottom" review of electronic voting systems certified for use in the State of California. Technical report, California Secretary of State, 2007. Available at `http://www.sos.ca.gov/voting-systems/oversight/top-to-bottom-review.htm`.

[6] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM (CACM)*, 24(2):84–90, Feb. 1981.

[7] S. Delaune, S. Kremer, and M. Ryan. Coercion-Resistance and Receipt-Freeness in Electronic Voting. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW '06)*, pages 28–39, Venice, Italy, July 2006.

[8] S. Delaune, S. Kremer, and M. Ryan. Verifying Properties of Electronic Voting Protocols. In *Proceedings of the 2006 IAVoSS Workshop On Trustworthy Elections (WOTE '06)*, pages 45–52, Cambridge, UK, June 2006.

[9] S. Delaune, S. Kremer, and M. Ryan. Verifying Privacy-type Properties of Electronic Voting Protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[10] D. Dolev and A. C.-C. Yao. On the Security of Public Key Protocols (Extended Abstract). In *22nd Annual Symposium on Foundations of Computer Science (FOCS '81)*, pages 350–357, Nashville, TN, USA, Oct. 1981.

[11] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology - Proceedings of CRYPTO '84*, pages 10–18, Santa Barbara, CA, USA, Aug. 1984.

[12] H. Hursti. Critical Security Issues with Diebold TSx, May 2006.

[13] Ireland Minister for the Environment, Heritage and Local Government John Gormley. Minister Gormley announces Government decision to end electronic voting and counting project. Available at `http://www.sos.state.oh.us/SOS/Text.aspx?page=4512`, 23 Apr. 2009.

[14] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach. Analysis of an Electronic Voting System. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy (S&P '04)*, pages 27–40, Berkeley, CA, USA, May 2004.

[15] S. Kremer, M. Ryan, and B. Smyth. Election Verifiability in Electronic Voting Protocols. In *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS '10)*, pages 389–404, Athens, Greece, Sept. 2010.

[16] S. Kremer and M. D. Ryan. Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In *Programming Languages and Systems: Proceedings of the 14th European Symposium on Programming (ESOP '05)*, volume 3444, pages 186–200, Edinburgh, Scotland, UK, Apr. 2005.

[17] C. Lambrinoudakis, D. Gritzalis, V. Tsoumas, M. Karyda, and S. Ikonomopoulos. Secure Electronic Voting: The Current Landscape. In D. Gritzalis, editor, *Secure Electronic Voting*, pages 101–122. Kluwer Academic Publishers, 2003.

[18] L. Loeber. E-Voting in the Netherlands; from General Acceptance to General Doubt in Two Years. In *Proceedings of the 3rd International Conference on Electronic Voting (EVOTE '08)*, pages 21–30, Castle Hofen, Bregenz, Austria, Aug. 2008.

[19] M. McGaley and J. P. Gibson. E-Voting: A Safety Critical System. Undergraduate thesis NUIM-CS-TR-2003-02, NUI Maynooth, Computer Science Department, 2003. Available at `http://www.cs.may.ie/research/reports/2003/index.html\#02`.

[20] R. Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, Cambridge, UK, 1999.

[21] National Institute of Standards and Technology. *FIPS Publication 180-3: Secure Hash Standard (SHS)*. United States Department of Commerce, Oct. 2008.

[22] Ohio Secretary of State Jennifer Brunner. Evaluation & Validation of Election-Related Equipment, Standards & Testing (EVEREST). Technical report, Ohio Secretary of State, 2007. Available at `http://www.sos.state.oh.us/SOS/Text.aspx?page=4512`.

[23] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM (CACM)*, 21(2):120–126, Feb. 1978.

[24] J. Rodrigues-Filho, C. Alexander, and L. Batista. E-Voting in Brazil - The Risks to Democracy. In *Proceedings of the 2nd International Workshop on Electronic Voting (EVOTE '06)*, pages 85–94, Castle Hofen, Bregenz, Austria, Aug. 2006.

[25] M. Ryan and B. Smyth. Applied Pi Calculus. In V. Cortier and S. Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*. IOS Press, Mar. 2011.

[26] C. Sturton, S. Jha, S. A. Seshia, and D. Wagner. On Voting Machine Design for Verification and Testability. In *Proceedings of the 2009 ACM Conference on Computer and Communications Security (CCS '09)*, pages 463–476, Chicago, IL, USA, Nov. 2009.

[27] D. Wagner, D. Jefferson, and M. Bishop. Security Analysis of the Diebold Accubasic Interpreter, Feb. 2006.

[28] S. Wolchok, E. Wustrow, J. A. Halderman, H. K. Prasad, A. Kankipati, S. K. Sakhamuri, V. Yagati, and R. Gonggrijp. Security Analysis of India's Electronic Voting Machines. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*, pages 463–476, Chicago, IL, USA, Oct. 2010.