http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 10 (2): 293-334, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.293.334

Refinement of Mechanized Proof of Security Properties of Remote Internet Voting Protocol in Applied PI Calculus with Proverif

Bo Meng
School of Computer, South-Center University for Nationalities,
MinYuan Road No. 708, HongShan Section, Wuhan, Hubei, 430074, China

Abstract: Secure remote internet voting protocols play an important role in electronic government. In order to assess its claimed security, several formal models of soundness and coercion-resistance have been proposed in the past literatures, but these formal models are not supported by mechanized tools. Recently, Backes *et al.* propose a new mechanized formal model of security properties including soundness and coercion-resistance in applied PI calculus. Acquisti protocol is one of the most important remote internet voting protocols that claims to satisfy formal definitions of key properties without strong physical constrains. But in the study the analysis of its claimed security is finished by hand. Owning to the contribution of Backes *et al.*, Acquisti protocol can be analyzed with mechanized tool. In this study, firstly the review of the formal analysis of electronic voting protocols are introduced we can find that the formal model and analysis of security properties mainly focus on receipt-freeness and coercion-resistance. Until now the security analysis model based on computational model have not been proposed; then applied PI calculus and the mechanized proof tool ProVerif are examined. After that Acquisti protocol is modeled in applied PI calculus. Finally security properties, including soundness and coercion resistance, are proved with ProVerif, a resolution-based mechanized theorem prover for security protocols. The result we obtain is that Acquisti protocol has the soundness and coercion-resistance in some conditions. To our best knowledge, the first mechanized proof of Acquisti protocol for an unbounded number of honest and corrupted voters is presented.

Key words: Mechanized tool, formal proof, soundness, coercion-resistance, ProVerif

INTRODUCTION

With the development of Internet and information technology, electronic government has got serious attention from enterprise and academic world. Owning to advantages of remote internet voting, it plays an important role in electronic government. In order to assess its security and increase confidence of the voters in remote internet voting system and protocols, many researchers have pay attention to design and verification on secure remote internet voting systems and protocols. Remote internet voting protocol is a key part of internet voting system. So how to develop and verify a practical secure internet voting protocol is a challenging issue (Abadi and Gordon, 1999).

The practical secure remote internet voting protocol should include the basic and expanded properties. Basic properties include privacy, completeness, soundness (Abadi and Fournet, 2001), fairness and invariableness. Expanded properties include universal verifiability (Sako and Killian, 1995), receipt-freeness (Benaloh and Tuinstra, 1994,) and coercion-resistance (Juels and Jakobsson, 2002). Recently research focus on implementation and formal analysis of receipt-freeness and coercion-resistance.

Soundness is typically consists of inalterability, eligibility and unreusability (Backes *et al.*, 2008a). Universal verifiability describes that any one can verify the fact that the election is fair and the published tally is correctly computed from the ballots that were correctly cast (Sako and Killian, 1995). Receipt-freeness is to protect against vote buying (Benaloh and Tuinstra, 1994). The voter can not produce a receipt to prove that he votes a special ballot. Coercion resistance means that it should offer not only receipt-freeness, but also defense against randomization, forced-abstention and simulation attacks (Juels and Jakobsson, 2002).

In the last twenty years many remote internet voting protocols (Benaloh and Tuinstra, 1994; Magkos *et al.*, 2001; Juels and Jakobsson, 2002; Acquisti, 2004; Chaum, 2004; Juels *et al.*, 2005; Chaum *et al.*, 2005; Rivest, 2006; Cichon *et al.*, 2008; Clarkson *et al.*, 2008; Meng, 2007a, 2009a, c, d; Meng *et al.*, 2010a-c; Meng and Wang, 2010), claimed on their security, have be proposed. In order to assess and verify security properties of remote internet voting protocol there are two model can be used:

one is formal model (or Dolev-Yao, symbolic model) in which cryptographic primitives are ideally abstracted as black boxes, the other is computational model (or cryptographic model) based on complexity and probability theory. Firstly each model formally defines security properties expected from security protocol and then develop methods for strictly proving that given security protocols satisfy these requirements in adversarial environments. Computational model is complicated and is difficult to get the support of mechanized proof tools. In contrast, symbolic model is considerably simpler than the computational model, proofs are therefore also simpler and can sometimes benefit from mechanized proof tools support. For example: SMV (McMillan, 1992; Mei *et al*., 2009), NRL (Meadows, 1996), Casper (Lowe, 1998), Isabelle (Paulson, 1998), Athena (Song, 1999), Revere (Kindred, 1999), SPIN (Maggi and Sisto, 2002), Brutus (Clarke *et al*., 2000), ProVerif (Blanchet, 2001), Scyther (Joseph and Cremers, 2006).

ProVerif (Blanchet, 2001) is an mechanized proof of cryptographic protocol verifier based on a representation of the protocol by Horn clauses or applied PI calculus. It can handle many different cryptographic primitives, including shared- and public-key encryption and signatures, hash functions and Deffie-Hellman key agreements, specified both as rewrite rules and as equations. It can also deal with an unbounded number of sessions of the protocol (even in parallel) and an unbounded message space. When ProVerif cannot prove a property, it can reconstruct an attack, that is, an execution trace of the protocol that falsifies the desired property. This verifier can prove the following properties: secrecy, authentication and more generally correspondence properties, strong secrecy, equivalences between processes that differ only by terms. ProVerif has been tested on protocols of the literature with very great results (http://www.proverif.ens.fr/proverif-users.html).

Acquisti (2004) protocol is one of the most important remote internet voting protocols that claims to satisfy formal definitions of key properties, such as soundness, individual verifiability, as well as receipt-freeness and coercion resistance without strong physical constrains. In their study, analysis of security properties of Acquisti protocol is done by hand, this method depend on experts knowledge and skill and is prone to make mistakes, so here we use mechanized proof tool ProVerif to verify security properties of Acquisti protocol.

The main contributions of this study are summarized as follows:

- Review the formal analysis of security properties in electronic voting protocol. Many formal models have been proposed, but only the Bakes *et al*. model supports the mechanized proof tool. The formal model and analysis of security properties mainly focus on receipt-freeness and coercion-resistance which are important properties. Until now the security analysis model based on computational model have not been proposed
- Apply the mechanized formal model proposed by Backes *et al*. (2008a) for mechanized proof of Acquisti protocol and its security properties. Therefore, Acquisti protocol is modeled in applied PI calculus and the soundness and coercion-resistance take into account. The proof itself is performed by mechanized proof tool ProVerif developed by Blanchet (2001)
- The result we obtain is that Acquisti protocol has coercion-resistance in some conditions. At the same time it also has the soundness. To our best knowledge, we are conducting the first mechanized proof of Acquisti protocol for an unbounded number of honest and corrupted voters

Formal methods are an important tool for designing and implementing secure cryptographic protocol. By applying techniques concerned with the construction and analysis of models and proving that certain properties hold in the context of these models, formal methods can significantly increase one's confidence that a protocol will meet its requirements in the real world.

The development of formal methods has started in 1980s (Yao, 1982; DeMillo *et al*., 1982; Dolev and Yao, 1983; Merritt, 1983; Blum and Micali, 1984; Hoare, 1985; Burrows *et al*., 1989, 1990). This field matured considerably in the 1990s. Some of the methods rely on rigorous but informal frameworks, sometimes supporting sophisticated complexity-theoretic definitions and arguments. Others rely on formalisms specially tailored for this task. Yet others are based on Mur (Mitchell *et al*., 1997) strand space (Thayer *et al*., 1998), SPI calculus (Abadi and Gordon, 1999) Kessler and Neumann logic (Kessler and Neumann, 1998) applied PI calculus (Abadi and Fournet, 2001).

Owning to the abstraction ideally of cryptography, formal methods are often quite effective; a fairly abstract view of cryptography often suffices in the design, implementation and analysis of security protocols. Formal methods enable relatively simple reasoning and also benefit from substantial work on proof methods and from extensive tool support, for example, SMV (McMillan, 1992), NRL (Meadows, 1996), Casper (Lowe, 1998), Isabelle (Paulson, 1998), Athena (Song, 1999), Revere (Kindred, 1999), SPIN (Maggi and Sisto, 2002), Brutus (Clarke *et al*., 2000), ProVerif (Blanchet, 2001), Scyther (Joseph and Cremers, 2006).

Delaune *et al.* (2006a) have done a path breaking work on the formal definition of receipt-freeness and coercion-resistance in applied PI calculus. Their formal model is based on Dolev-Yao model. They formalize receipt-freeness as an observational equivalence. The idea is that if the attacker can not find if arbitrary honest voters V_A and V_B exchange their votes, then in general he can not know anything about how V_A (or V_B) voted. This definition is robust even in situations where the result of the election is such that the votes of V_A and V_B are necessarily revealed. They also assume that the voter cooperates with the coercer by sharing secrets, but the coercer cannot interact with the voter to give her some prepared messages. They use adaptive simulation to formalize coercion-resistance. The ideas of this definition is that whenever the coercer requests a given vote then V_B can change his vote and counterbalance the outcome. However, avoid the case where $V' = V_A\ \{c/v\}^{c_1,c_2}$ letting V_B vote α is needed. Therefore, requirement that when we apply a context C, intuitively the coercer, requesting $V_A\ \{c/v\}^{c_1,c_2}$ to vote c, V' in the same context votes α. There may be circumstances where V' may need not to cast a vote that is not. In the case of coercion-resistance, the coercer is assumed to communicate with voter during the protocol and can prepare messages which she should send during the election process. Their formal definition of coercion-resistance base on the informal definition: a voter cannot cooperate with a coercer to prove to him that she voted in a certain way. Lee *et al.* (2003) protocol is analyzed with their formal model. Meng (2008) also apply their formal model to analyze the protocol (Meng, 2007a). Delaune *et al.* (2005) model receipt-freeness and analyze Lee *et al.* (2003) protocol. Delaune *et al.* (2006b) use applied PI calculus to model fairness, eligibility, privacy, receipt-freeness and coercion-resistance and analyze the protocols (Fujioka *et al.*, 1992; Lee *et al.*, 2003). Backes *et al.* (2008a) point out that definitions of coercion-resistance Delaune *et al.* (2006a) are not amenable to automation and do not consider forced-abstention attacks and do not apply to remote voting protocols, they give an formal model of security properties of remote internet voting protocol in applied PI calculus and use the ProVerif to mechanized verify the security properties of Juels *et al.* (2005) protocol. Gerling *et al.* (2008) apply the model (Backes *et al.*, 2008a) with ProVerif to mechanized verify Clarkson *et al.* (2008) protocol. Meng *et al.* (2010b, c) also use the Backes *et al.* (2008a) model to mechanized verify Acquisti (2004) protocol and Meng *et al.* (2010a) protocol.

Jonker and de Vink (2006) also point out that the formal model (Delaune *et al.*, 2006a) offers little help to identify receipts when receipts are present. So Jonker and de Vink present a new formal method, which uses the process algebra, to analyze receipts based on their informal definition: a receipt r is an object that proves that a voter v cast a vote for candidate c. This means that a receipt has the following properties: (R1) receipt can only have been generated by v. (R2) receipt proves that voter chose candidate. (R3) receipt proves that voter cast her vote. Jonker and de Vink provide a generic and uniform formalism that captures a receipt. Jonker and de Vink formal model is also simpler than Delaune *et al.* formal model. They use the formalism to analyze the voting protocols (Benaloh and Tuinstra, 1994; Sako and Killian, 1995; Hirt and Sako, 2000; Aditya *et al.*, 2004; Hubbers *et al.*, 2005). Meng (2007b) analyzes receipt-freeness of the protocols (Fujioka *et al.*, 1992; Cramer *et al.*, 1997; Juels and Jakobsson, 2002; Acquisti, 2004) based on formalism (Jonker and de Vink, 2006).

About definition of receipt proposed by Jonker and de Vink (2006) think it is worth discussing. Firstly, about (R1) r can only have been generated by v, in some voting protocol, one part of receipt is generated by the authority, not generated by voter. Secondly, they give the following auxiliary receipt decomposition functions: a: Rcpt ?AT, which extracts the authentication term from a receipt. Authentication term should be the identification of voter. Thirdly the author does not prove the generic and uniform formalism that is right in their study. Finally they use a special notation, it difficult to use and generalize it. Hence, Meng gives a formal logic framework for receipt-freeness based on Kessler and Neumann (1998) and apply it to analyze (Fujioka *et al.*, 1992) protocol.

Knowledge-based logics have been also used in the studies (Jonker and Pieters, 2006; Baskar *et al.*, 2007; Van Eijck and Orzan, 2007) to formally analyze the security properties of e-voting protocol. Jonker and Pieters (2006) formalize the concept of receipt-freeness from the perspective of a anonymity approach in epistemic logic which offers, among others, the possibility to write properties allowing to reason about the knowledge of an agent a of the system with respect to a proposition p. They classify receipt-freeness into two types: weak receipt-freeness and strong receipt-freeness. Weak receipt-freeness implies that the voter cannot prove to the spy that she sent message m during the protocol, where m is the part of a message representing the vote. Here, no matter what information the voter supplies to the spy, any vote in the anonymity set is still possible. In other words, for all possible votes, the spy still suspects that the voter cast this particular vote; or: the spy is not certain she did not cast this vote. Baskar *et al.* (2007) gave the formal definition of secrecy, receipt-freeness, fairness, individual verifiability based on knowledge-based logic and analyze receipt-freeness of Fujioka *et al.* (1992) protocol. Van Eijck and Orzan (2007) use dynamic epistemic Logic to model security protocols and properties, in particular anonymity properties. They apply it to Fujioka *et al.* (1992) scheme and find the three phases should be strictly separated, otherwise anonymity is compromised. Mauw *et al.* (2007)

use the process algebra to analyze the data anonymity of the voting scheme (Fujioka *et al*., 1992). Talbi *et al*. (2008) use ADM logic to specify security properties (fairness, eligibility, individual verifiability and universal verifiability) and analyze Fujioka *et al*. (1992) protocol. Their goal is to verify these properties against a trace-based model. Groth (2004) evaluated the voting scheme based on homomorphic threshold encryption with universal composability framework. He formalizes the privacy, robustness, fairness and accuracy.

The formal methods used in formal models of soundness, receipt-freeness and coercion-resistance are presented in Table 1. We can found in Table 1 until now only the Bakes *et al*. model supports the mechanized proof tool. The security properties formally modeled is presented in Table 2. The formally analyzing security properties in the Internet voting protocol is described in Table 3. From Table 1-3 we can get that the formal model and analysis of security properties mainly focus on receipt-freeness and coercion-resistacne that are important properties.

Table 1: The formal methods used in formal models of soundness, receipt-freeness and coercion-resistance

Properties	Formal method	Delaune *et al.* (2006a)	Jonker and de Vink (2006)	Meng (2009b)	Jonker and Pieters (2006)	Baskar *et al.* (2007)	Bakes *et al.* (2008a)
Soundness	applied PI calculus						•○
Receipt-freeness	Applied PI calculus	•					•○
	Process algebra		•				
	Kessler and Neumann logic			•			
	Epistemic logic				•		
	Knowledge-based logic					•	
Coercion-resistance	Applied PI calculus	•					•○

The mark • represents the formal method is used. The mark ○ represents the formal models is supported by mechanized proof tool

Table 2: The security properties formally modeled

Properties	Fairness	Soundness	Eligibility	Privacy	Receipt-freeness	Coercion-resistance	Secrecy	Individual verifiability	Universal verifiability	Anonymity
Baskar *et al.* (2007)	•				•		•	•		
Meng (2009b)					•					
Jonker and de Vink (2006)					•					
Delaune *et al.* (2005)					•					
van Eijck and Orzan (2007)										•
Kremer and Ryan (2005)	•		•	•						
Delaune *et al.* (2006b)	•		•	•	•	•				
Bakes *et al.* (2008a)		•○	•○		•○	•○				
Talbi *et al.* (2008)	•		•					•	•	
Mauw *et al.* (2007)										•

The mark • represents the property is formally defined; The mark ○ represents the formal definitions is supported by mechanized proof tool

Table 3: Formally analyzing security properties in the Internet voting protocol

	Analyzed protocol	Soundness	Receipt-freeness	Concerion-resistance
Bakes *et al.* (2008a)	Juels *et al.* (2005)	•	•	•
Gerling *et al.* (2008)	Clarkson*et al.* (2008)	•	•	•
Meng *et al.* (2010c)	Meng *et al.* (2010a)	□	•	•
Baskar *et al.* (2007)	Fujioka *et al.* (1992)		□	
Meng (2009b)	Fujioka *et al.* (1992)		□	
	Meng (2007a)		•	
Meng (2007b)	Fujioka *et al.* (1992)		□	
	Cramer *et al.* (1997)		□	
	Juels and Jakobsson (2002)		Δ	
	Acquisti (2004)		Δ	
Jonker and de Vink (2006)	Benaloh and Tuinstra (1994)		□	
	Sako and Kilian (1995)		□	
	Hirt and Sako (2000)		•	
	Aditya *et al.* (2004)		•	
	Hubbers *et al.* (2005)		★	
Delaune *et al.* (2005)	Lee *et al.* (2003)		★	

	Analyzed protocol	Fairness	Eligibility	Privacy	Receipt-freeness	Coercion-resistance	Individual verifiability	Universal verifiability	Anonymity
Van Eijck and Orzan (2007)	Fujioka *et al.* (1992)								•
Mauw *et al.* (2007)									•
Talbi *et al.* (2008)		•	•				•	★	
Delaune *et al.* (2005)		•	•	•					
Meng (2008)	Meng (2007a)				•	•			
Delaune *et al.* (2006b)	Lee *et al.* (2003)	•	•	•	•	•			
	Fujioka *et al.* (1992)	•	•	•	★	★			

The mark • represents the protocol has the property; The mark □ represents the protocol has not the property; The mark Δ represents the protocol has the property with some condition

Above previous formal models and analysis is based on symbolic model. Until now people have not proposed a security analysis model based on computational model.

CONTRIBUTION AND OVERVIEW

In the last two decades many remote internet voting protocol have been introduced. Owning to the complexity how to assess their security is a challenging issue. Formal method is crucial to assess their security. So, in this study we firstly review the development of the formal method on remote electronic voting protocol, we found that several formal model have been proposed, but only the Bakes *et al.* model support the mechanized proof tool and the formal model and analysis of security properties mainly focus on receipt-freeness and coercion-resistance that are important properties. Until now people have not proposed a security proof model based computational model; and then apply the mechanized formal model proposed by Backes *et al.* (2008a) to prove Acquisti protocol and its security properties including soundness and coercion-resistance. Therefore, first, Acquisti protocol is modeled in applied PI calculus and then its proof is performed by mechanized proof tool ProVerif. The result is that Acquisti protocol has the soundness. At the same time it has also coercion-resistance in the conditions that the channel between registration authority and voter is private. To our best knowledge, we are conducting the first mechanized proof of Acquisti protocol for an unbounded number of honest and corrupted voters.

Acquisti protocol is modeled with applied PI calculus (Abadi and Fournet, 2001). Our choice is based on the fact that applied PI calculus allows the modeling of relations between data in a simple and precise manner using equational theories over term algebra. The general analysis model is introduced in Fig. 1. Acquisti protocol in applied PI calculus is illustrated in Fig. 2. There, the security properties model is equivalence between processes, while the attacker is thought as an arbitrary process running in parallel with the protocol process representing the adversary model, which is the parallel composition of the (sequential) protocol participants' processes. The considered attacker is stronger than the basic Dolev_Yao attacker since it can exploit particular relations between the messages by using particular equational theories stating the message relations.

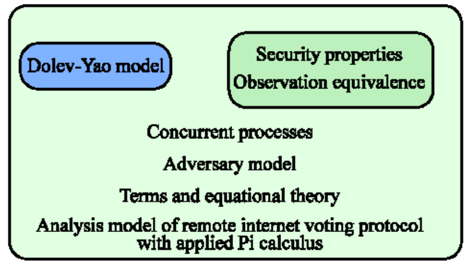


Fig. 1: Analysis model of remote internet voting protocol with applied PI calculus

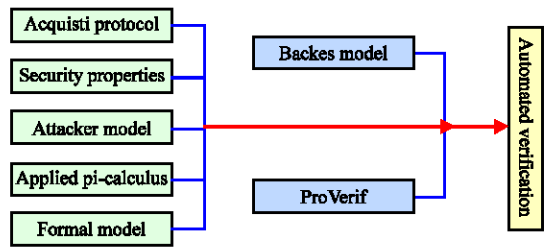


Fig. 2: Mechanized proof of Acquisti protocol

REVIEW OF THE APPLIED PI CALCULUS

Applied PI calculus is a language for describing concurrent processes and their interactions based on Delov_Yao model. Applied PI calculus is an extension of the PI calculus that inherits the constructs for communication and concurrency from the pure pi-calculus. It preserves the constructs for generating statically scoped new names and permits a general systematic development of syntax, operational semantics equivalence and proof techniques. At the same time there are several powerful mechanized proof tool supported applied pi-calculus, for example, ProVerif. Applied PI calculus with ProVerif has been used to study a variety of complicated security protocols, such as a certified email protocol, just fast keying protocol (Abadi *et al.*, 2004; Juels *et al.*, 2005) remote electronic voting protocol (Backes *et al.*, 2008a), a key establishment protocol, direct anonymous attestation protocol (Backes *et al.*, 2008b), TLS protocol (Bhargavan *et al.*, 2008; Meng *et al.*, 2010a) remote internet voting protocol (Meng *et al.*, 2010c).

Syntax: In applied PI calculus, terms in Fig. 3 consists of name, variables and signature Σ. Σ is set of function symbols, each with an arity. Terms and function symbols are sorted and of course function symbol application must respect sorts and arities. Typically, we let a, b and c range over channel names. Let, x, y and z range over variables and u over variables and names. We abbreviate an arbitrary sequence of terms $M_1,\cdots,M_l$ to $\widetilde{M}$.

$M,N,T,V::=$		terms
	x	variable
	$a,b,c,...,m,n$	name
	$f(M_1,L,M_l)$	function application

Fig. 3: Terms

$P,Q,R::=$		plain processes
	0	null process
	$Q\|P$	parallel composition
	$!P$	replication
	$vn.P$	name restriction
	if $M=N$ then P else Q	conditional
	$in(u,x).P$	message input
	$out(u,N).P$	message output

Fig. 4: Plain process

$A,B,C::=$		extended processes
	P	plain process
	$A\|B$	parallel composition
	$vn.A$	name restriction
	$vx.A$	variable restriction
	$\{^M/_x\}$	active substituation

Fig. 5: Extended process

In applied PI calculus, it has plain processes and extended processes. Plain processes in Fig. 4 are built up in a similar way to processes in the PI calculus, except that messages can contain terms (rather than just names) and that names need not be just channel names:

The process 0 is an empty process. The process Q|P is the parallel composition of P and Q. The replication !P produces an infinite number of copies of P which run in parallel. The process vn.P firstly creates a new, private name then executes as P. The abbreviation $v\tilde{n}$ is a sequence of name restrictions $v_{n1},...,v_{nl}$. The process in (u,x).P receives a message from channel u and runs the process P by replacing formal parameter x by the actual message. We use $in(u,\tilde{M}).P$ for the input of terms $M_1,...,M_l$. The process out(u,N).P is firstly ready to output the message N on the channel u and then runs the process P. The process $out(u,\tilde{N}).P$ is the abbreviation for the output of terms $N_1,...,N_l$. The conditional construct if M = N then P else Q runs that if M and N are equal, executes P, otherwise executes Q.

Extended processes in Fig. 5 add active substitutions and restriction on variables:

We write {M/x} for active substitution which replaces the variable x with the term M. The substitution typically appears when the term M has been sent to the environment, but the environment may not have the atomic names that appear in M; the variable x is just a way to refer to M in this situation. We write fv(A), fn(A) for the free variables and name in a process A, respectively. We write bv(A) ,bn(A) for the bound variables and name in a process A, respectively.

A frame is an extended process built up from 0 and active substitutions of the form {M/x} by parallel composition and restriction. Let, φ and ψ range over frames. The domain dom(φ) of a frame φ is the set of the variables that φ exports. Active substitutions are useful because they allow us to map an extended process A to its frame φ(A) by replacing every plain process in A with 0. The frame φ(A) can be viewed as an approximation of A that accounts for the static knowledge A exposes to its environment, but not A's dynamic behavior. The domain of dom(A) is the domain of φ(A). A process or extended process with a hole is called a context. The plain process with a hole is called plain context. Those plain contexts without replications, conditionals, inputs or outputs are called sequential contexts. A context C[_] closes A if C[A] is closed.

A signature Σ is equipped with an equational theory that is an equivalence relation on terms that is closed under substitutions of terms for variables. An equational theory is generated from a finite set of equational axioms. It models the algebraic properties of cryptographic primitives. We write Σ|-M = N for equality within the equational theory Σ and Σ|-/M = N for inequality.

Operational semantics: The operational semantic is inherited from the applied PI calculus and is defined by structural equivalence (≡) and internal reduction (→).

Structural equivalence in Fig. 6 (≡) is the smallest equivalence relation on extended processes that is closed by α conversion on both names and variables, by application of evaluation contexts, Structural equivalence can make the introduction and application of an active substitution and the equational rewriting of the terms in a process. Structural equivalence satisfies the rules in the following:

The rules for parallel composition and restriction are standard. A_{LIAS} enables the introduction of an arbitrary active substitution. S_{UBST} describes the application of an active substitution to a process that is in contact with it. Rewrite deals with equational rewriting.

Internal reduction (→) relies on the equational theory and defines the semantics of process conditionals as well as input and output. Internal reduction in Fig. 7 is the smallest relation on extended processes closed by structural equivalence and application of evaluation contexts such that:

T_{HEN} and E_{LSE} directly depend on the underlying equational theory; using E_{LSE} sometimes requires that active substitutions in the context be applied first, to yield ground terms M and N.

We write $A \Downarrow a$ when A can send a message on a, that is, when $A \to^{*} C[\bar{a}\langle M\rangle.P]$ for some evaluation context C[_] that does not bind a. Observational equivalence constitutes an equivalence relation that captures the equivalence of processes with respect to their dynamic behavior. Observational equivalence ($\approx$) is the largest symmetric relation R between closed extended processes with the same domain such that A R B implies:

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	$n.0 \equiv 0$
NEW-C	$\nu u.\nu v.A \equiv \nu v.\nu u.A$
NEW-PAR	$A \mid \nu u.B \equiv \nu u.(A \mid B)$ when $u \notin fv(A) \cup fn(A)$
ALIAS	$\nu x.\{M/x\} \equiv 0$
SUBST	$\{M/x\} \mid A \equiv \{M/x\} \mid A\{M/x\}$
REWRITE	$\{M/x\} \equiv \{n/x\}$ when $\sum \mid -M = N$

Fig. 6: Structural equivalence

COMM	$\bar{a}\langle x\rangle.P \mid a(x).Q \to P \mid Q$
THEN	if $M = M$ then P else $Q \to P$
ELSE	if $M = N$ then P else $Q \to Q$
	for any ground terms M and N such that $\sum \mid - / M = N$

Fig. 7: Internal reduction

1. if $A \Downarrow a$, then $B \Downarrow a$;
2. if $A \to^{*} A'$, then $B \to^{*} B'$ and A' R B' for some B';
3. $C[A]$R $C[B]$ for all closing evaluation contexts C[_] .

MECHANIZED PROOF TOOL PROVERIF

ProVerif is a mechanized cryptographic protocol verifier based on a representation of the protocol by Horn clauses or applied PI calculus. It can handle many different cryptographic primitives, including shared- and public-key cryptography (encryption and signatures), hash functions and Deffie-Hellman key agreements, specified both as rewrite rules and as equations. It can also deal with an unbounded number of sessions of the protocol (even in parallel) and an unbounded message space. When ProVerif cannot prove a property, it can reconstruct an attack, that is, an execution trace of the protocol that falsifies the desired property. ProVerif can prove the following properties: secrecy, authentication and more generally correspondence properties, strong secrecy, equivalences between processes that differ only by terms. ProVerif has been tested on protocols of the literature with very encouraging results (http://www. proverif.ens.fr/proverif-users.html). Recent research came up with an abstraction of zero-knowledge proofs, a primitive heavily used within electronic voting protocols such as Juels *et al.* (2005), Meng *et al.*, (2010a) and Clarkson *et al.* (2008) protocols that is accessible to an mechanized proof using ProVerif (Backes *et al.*, 2008a; Gerling *et al.*, 2008; Meng *et al.*, 2010c).

ProVerif in Fig. 8 is for the proof of trace-based security properties and observational equivalence. Since, the security definitions for basic properties and

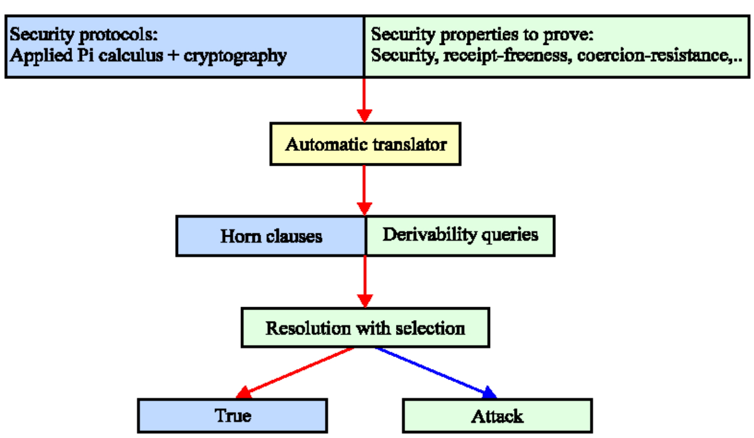


Fig. 8: Mechanized proof tool ProVerif (Blanchet, 2008)

expanded properties including receipt-freeness and coercion-resistance for Backes *et al.* (2008b) model heavily rely on observational equivalences, ProVerif is the only tool for our purpose of an mechanized proof of Acquisti protocol. Inspired by study of Backes *et al.* (2008a) and Gerling *et al.* (2008), we use it to mechanized prove Acquisti protocol.

BACKES *ET AL.* (2008a) MODEL

Here, we describes Backes *et al.* (2008a) model used to mechanized prove Acquisti protocol. Backes *et al.* (2008a) model formalize key properties including the soundness, receipt-freeness and coercion-resistance in remote internet voting protocol with applied PI calculus. Backes *et al.*, 2008a model mainly model the soundness, receipt-freeness and coercion-resistance. In Backes *et al.* (2008a) model the voter are classified into three types of voter: honest voter, corrupted voter and ad-hoc voter. Honest voter are issued an identity by an issuer authority and behave according to the protocol specification. Corrupted voter will register and then simply output all their registration credentials on a public channel, thus the coercer and vote buyer can impersonate him in order to mount any sort of attack. Ad-hoc voters can behave arbitrarily; they do not necessarily follow the protocol, but are also not necessarily corrupted. In the following section we first introduce the soundness including inalterability, eligibility and unreusability, then receipt-freeness and coercion-resistance in Backes *et al.* (2008a) model.

Soundness:

- **Informal definition:** In the study (Backes *et al.*, 2008a), soundness is typically consists of the following three separate properties:
  - **Inalterability:** No one can change anyone else's vote
  - **Eligibility:** Only eligible voters are able to vote
  - **Unreusability:** Every voter can vote only once

Backes *et al.* (2008a) model formalizes the definition of soundness with the events including beginvote(id,v), endvote(v), startid(id) and startcorid(id). The events in the soundness property are also used later in the modeled processes. beginvote(id, v) starts the voting phase for a voter with id and the intention to vote for v whereas endvote(v) is the tallying of this vote. startid(id) and startcorid(id) indicate the start of the registration phase for an eligible voter or an corrupted voter with id.

- **Formal definition:** A trace t guarantees soundness if and only if the following conditions hold in Fig. 9

Condition 1a and b models the inalterability and eligibility, respectively. This is done through requiring that every counted vote is either a vote casted by an eligible voter or a corrupted voter. In order to achieve unreusability it has to be assured that the matching between endvote and beginvote(id, v) is injective. This is ensured by Condition 1a in combination with condition 2.

Receipt-freeness:

- **Informal definition:** The voter can not produce a receipt to prove that he votes a special ballot. Its purpose is to protect against vote buying. This definition thus refers to an attacker that does not try to vote by impersonating the voter, but just tries to get a proof that the voter voted in a special way
- **Formal definition:** An election context S is receipt-freeness if there exists a plain process V' such that in Fig. 10

1. for any t_1, t_2, v such that $t = t_1 :: \text{endvote}(v) :: t_2$, there exists id, t', t'', t''' such that :
(a) $t_1 = t' :: \text{stardid}(id) :: t'' :: \text{beginvote}(id, v) :: t'''$ and $t' :: t'' :: t''' :: t_2$ guarantees soundness;
(b) or $t_1 = t' :: \text{stardcorid}(id) :: t''$, and $t' :: t'' :: t^2$ guarantees soundness

2. for any t_1, t_2, id such that $t = t_1 :: \text{startid}(id) :: t_2$ or $t = t_1 :: \text{stardcorid}(id) :: t_2$, the events startid(id) and startcorid(id) do not occur in $t_1 :: t_2$

An annotated election process EP guarantees soundness if and only if all its possible traces guarantee soundness.

Fig. 9: Formal definition of soundness

$$1.\ \nu c.\left(!c(x) \mid V'\right) \approx \text{let } x_{id} = i \text{ in } V^{reg}\left[\text{let } x_v = v' \text{ in } V^{vote} \mid V^{fake}\left[\text{let } x_v \in \tilde{v} \text{ in } V^{vote}\right]\right]$$

$$2.\ S\left[V^{receipt(c)}(v) \mid V_j(v') \mid V_k^{inv-reg}\right] \approx S\left[V' \mid V_j(v) \mid V_k^{abs}\right]$$

Fig. 10: Formal definition of receipt-freeness

Condition 1 models that the voter V' does not only vote v' as a regular voter, but additionally uses V^{fake} to generate fake secrets, casts an extra vote using them and provides a receipt of this invalid voting. Condition 2 deal with that an additional voter k that votes with fake registration secrets in case the voter i complies with the request of the coercer and simply abstains if i cheats the vote buyer by casting a vote with fake secrets.

Coercion-resistances:

- **Informal definition:** A coercion-resistant voting protocol should offer not only receipt-freeness, but also defense against randomization, forced-abstention and simulation attacks
- **Receipt-freeness:** The voter can not produce a receipt to prove that he votes a special ballot
- **Immunity to randomization attack:** The idea is for an attacker to coerce a voter by requiring that she submit randomly composed balloting material. The effect of the attack is to nullify the choice of the voter
- **Immunity to forced-abstention attack:** This is an attack related to the previous one based on randomization. In this case, the attacker coerces a voter by demanding that she refrain from voting
- **Immunity to simulation attack:** An attacker coerce voters into divulging private keys or buying private keys from voters and then simulating these voters at will, i.e., voting on their behalf

The formalization that encompasses all properties except randomization attacks depicted below is taken from Backes *et al*. (2008a) model.

In order to formalize coercion-resistance, the process called Extractor is introduced. Extractor plays an important role in formalization of coercion-resistance, which extracts the vote the coercer casts on behalf of Extractor and tallies it directly. Extractor depends on the construction of the particular electronic voting protocol and has to be provided by the user.

- **Definition of Extractor:** A context $E_k^{c_1,c_2,z}$ is an Extractor if and only if:

$$E_k^{c_1,c_2,z} = \text{ let } x_{id} = k \text{ in } V^{reg}\left[\nu\tilde{m}.\left(c_1(x).P_1 \mid !c_2(y).P_2 \mid C\left[\text{if } z \in \tilde{v} \text{ then } [\ ]\right]\right)\right]$$

For some plain processes P_1, P_2 and a sequential context C such that $c_1, c_2 \not\subset fn(P_1) \cup fn(P_2) \cup fn(C)$, $z \in captured(C)$, all inputs and outputs in P_1, P_2, and C occur on the private channels in $\tilde{m}$, and such channels are never output.

The channels c_1 and c_2 are the channels shared by the extractor with the coerced voter and the tallying authority, respectively. If the coercer casts a vote, then the variable z should hold this vote. The context C is required to be sequential so it does not contain any replications, which means that:

$$E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle z\rangle\right]$$

can tally at most one vote.

- **Formal definition:** An election context S guarantees coercion-resistance if there exist channels c, c_1 and c_2, a sequential process V^{fake}, an Extractor $E^{c1,c2,z}$ and an election context S', such that in Fig. 11

In condition 1 (hypothesis) a modified election context s' is used that only differs from s in that the tallying authority additionally outputs messages on the channel c_2 shared with Extractor. In condition 2 the left side process contains the voter V_i that is in accordance with the orders of the coercer, running in parallel with the voter V_j casting a vote v' and the process $E^{c1,c2,z}[0]$, that is intuitively equivalent to a voter nullifying her vote. In the right side election process the voter V_i cheats the coercer by providing him with fake registration secrets and then votes v', the voter V_j participates in the registration phase and then abstains and the extractor process:

$$E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle z\rangle\right]$$

tallies the vote the coercer casts on behalf of v_i. In condition 3 if the cheated coercer abstains, then the Extractor needs to abstain as well; In condition 4 if the cheated coercer casts a valid vote using the fake registration secrets he received from v_i, the Extractor

1. there exist an election context S'' and two authority process A, A'
such that $S = S''\left[A \middle| [\ ]\right]$, $S' \equiv \nu c_1, c_2.S''\left[A' \middle| [\ ]\right]$, and $\nu c_2.\left(A \middle| !c_2(x)\right) \approx A$;

2. $S'\left[V_i^{\text{coerced}(c,c_1)} \middle| V_j(v') \middle| E_k^{c_1,c_2,z}[0]\right] \approx S'\left[V_i^{\text{cheat}(c,c_1)}(v') \middle| V_j^{\text{abs}} \middle| E_k^{c_1,c_2,z}\left[\overline{c_{\text{votes}}}\langle z\rangle\right]\right]$
where $v' \in \tilde{v}$ is a valid vote;

3. $\nu c.S'\left[!c(x) \middle| V_i^{\text{cheat}(c,c_1)}(v') \middle| V_j^{\text{abs}} \middle| E_k^{c_1,c_2,z}\left[\overline{c_{\text{votes}}}\langle z\rangle\right]\right] \approx S\left[V_i(v') \middle| V_j^{\text{abs}} \middle| V_k^{\text{abs}}\right]$;

4. let $P = c(\tilde{x}).\text{let } x_v = v \text{ in } V^{\text{vote}}\left\{\tilde{x}/\tilde{u}\right\}$, $v \in \tilde{v}, \tilde{u} = \text{captured}\left(V^{\text{reg}}\right)$,
and $\tilde{x} \cap \tilde{u} = \varnothing$ then $\nu c.S'\left[P \middle| V_i^{\text{cheat}(c,c_1)}(v') \middle| V_j^{\text{abs}} \middle| E_k^{c_1,c_2,z}\left[\overline{c_{\text{votes}}}\langle z\rangle\right]\right] \approx c.S'\left[P \middle| V_i^{\text{cheat}(c,c_1)}(v') \middle| V_j^{\text{abs}} \middle| E_k^{c_1,c_2,z}\left[\overline{c_{\text{votes}}}\langle v\rangle\right]\right]$;

5. $S\left[V_i^{\text{inv-reg}}\right] \approx \nu c_{\text{votes}}.\left(!c_{\text{votes}}(x)\right) \middle| S\left[V_i(v)\right]$, where v is a valid vote.

Fig. 11: Formal definition of coercion-resistance

needs to tally precisely this vote. In condition 5 an additional restriction is introduced that justifies the abstraction of the third voter by the Extractor: votes with invalid registration secrets are silently discarded by the tallying authority. If this was not the case a coercer could easily distinguish real from fake registration secrets.

ACQUISTI PROTOCOL

Acquisti (2004) protocol promises that it can protect voters' privacy and achieves universal verifiability, receipt-freeness and coercion-resistance without ad hoc physical assumptions or procedural constraints. It mainly applies threshold Paillier cryptosystem (Paillier, 1999), bulletin board that is a public broadcast channel with memory where a party may write information that any party may read, Mix net that guarantees privacy is a distributed protocol that takes as input a set of messages and returns an output consisting of the re-encrypted messages permuted according to a secret function (Chaum, 1981), proof of knowledge that two ciphertexts are encryption of the same plaintext (Baudron *et al*., 2001), designated verifier Proof of knowledge (Jakobsson *et al*., 1996; Hirt and Sako, 2000). Acquisti assumes that the private key is private and that an attacker cannot control every possible communication between the voter and an authority.

In Acquisti protocol there are five entities: registration authority, issue authority, bulletin board, voters, tallying authority. Registration authority is responsible for authenticating the voters. Issue authority takes charge of issuing the related key and credentials. Voters register for voting, get their credentials and post a vote. Tallying authority is responsible for tallying ballots. Model of Acquisti protocol is described in Fig. 12.

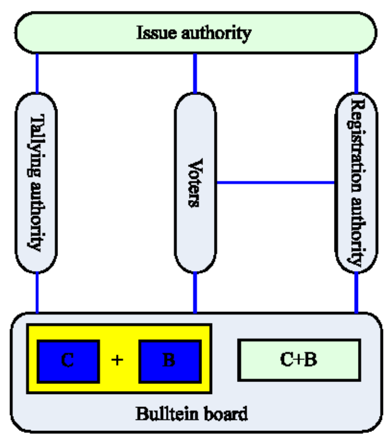


Fig. 12: Model of Acquisti protocol

Acquisti protocol consists of preparation phase, voting phase and tallying phase. In preparation phase the related keys and ballot are generated. Issuer authority creates the voting credential shares and posts copies of the shares of credentials encrypted with Paillier cryptosystem to a bulletin board. The same credential shares encrypted with different Paillier public keys and attach a designated verifier proof of the equivalence between the encrypted share and the one the voter has received to its message are also provided to voters. Issuer authority also creates the ballots shares which are encrypted with the two different Paillier public keys. Both the sets of encrypted ballots shares are posted on the bulletin board together with zero-knowledge proofs that

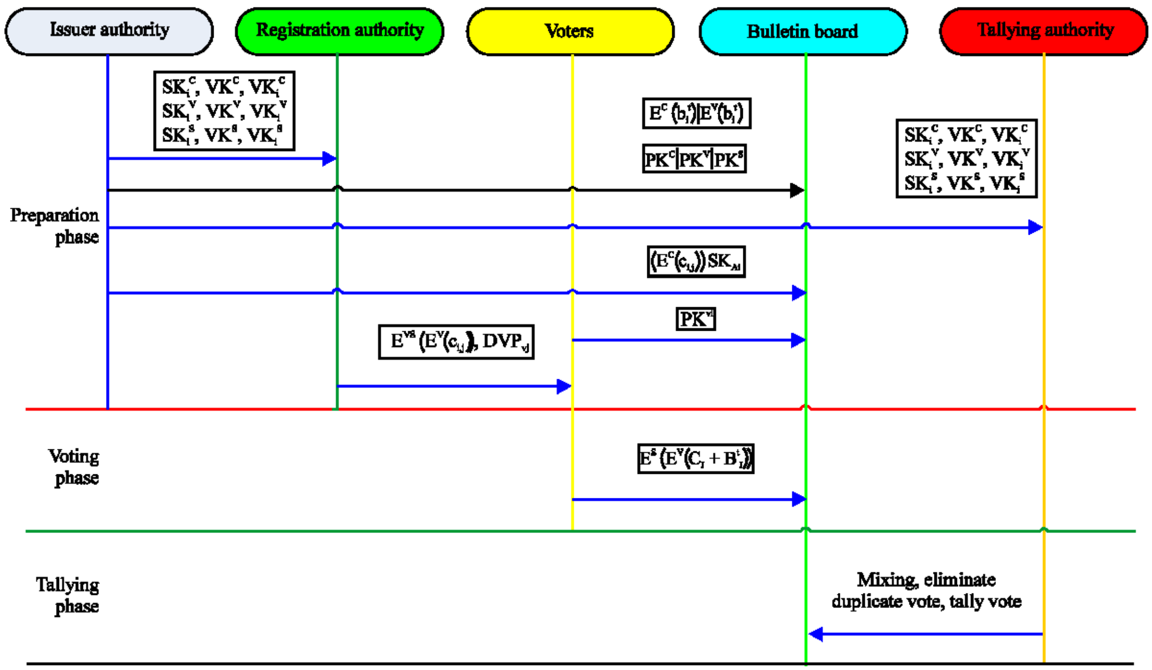


Fig. 13: The structure of message of Acquisti protocol

each pair of ciphertexts are encryptions of the same ballot share and are then signed by issuer Authority. In voting phase the voter vote his favor ballot and post it to bulletin board. Each voter multiplies the shares she has received from issuer authority together with the encrypted shares of the ballot. Because of the homomorphic properties of Paillier cryptosystems, the resulting ciphertext includes the sum of those shares and the ballot's shares. The resulting ciphertext is sent to the bulletin board. In the last phase, tallying phase, the tallying authority tallies the ballot and publishes the result in bulletin board. The structure of message is described in Fig. 13.

Preparation phase: Every issue authority $A_i(i = 1......l)$ creates l random numbers c as $c_{i,j}$, representing shares of credentials, for each eligible voter $veter_j(l = 1......l)$. For each $c_{i,j}$, A_i performs two operations: first, it encrypts $c_{i,j}$ using PK^C and appropriate secret randomization, signs the resulting ciphertext with SK_i^C and publishes it on bulletin board on a row publicly reserved for the shares of credential of voter:

$$v_j : \left(E^C\left(c_{i,j}\right)\right)SK_{A_i}$$

SK_{A_i} represents the signature of authority A_i Second, A_i also encrypts $c_{i,j}$ using PK^V and appropriate secret randomization, without signing it, but attaching to it a designated verifier proof DVP_{v_j} of equality of plaintexts $E^C(c_{i,j})$ and $E^V(c_{i,j})$. The proof is designated to be verifiable only by $voter_j$. A_i encrypts this second message with $voter_j$'s public key and sends it:

$$voter_j : E^{v_j}\left(E^V\left(c_{i,j}\right), DVP_{v_j}\right)$$

E^{voter_j} represents RSA encryption under $voter_j$'s public key.

Voting phase: For each encrypted share of credential she receives, $voter_j$ verifies the designated verifier proof of equality between $E^V(c_{i,j})$ and the corresponding $E^V(c_{i,j})$ that has been signed and published in her reserved area of bulletin board. Upon successful verification, she multiplies together the shares $E^V(c_{i,j})$.

$$\prod_{j=j,i=1,\cdots,s}\left(E^V\left(c_{i,j}\right)\right) = E^V\left(\sum_{j=j,i=1,\cdots,s} c_{i,j}\right) \equiv E^V\left(C_j\right)$$

C_j is the sum of the various shares of credentials. $voter_j$. Voter chooses the ballot shares $E^V(b_1^t), \cdots, E^V(b_s^t)$, generates:

$$E^V(C_j)E^V\left(B_j^t\right) = E^V\left(\sum_{i=1,\cdots,s} c_{i,j} + \sum_{i=1,\cdots,s} b_{i,j}^t\right) \equiv E^V(C_j + B_j^t)$$

and sends $E^S\left(E^V(C_j + B_j^t)\right)$ to bulletin board.

Tallying phase: After the voting time expires, all ballots on bulletin board posted by allegedly eligible voters are mixed by the tallying authorities. The shares of credentials posted by the registration authorities are also combined and then mixed. Tallying authorities thus obtain two lists: a list of encrypted, mixed credentials the registration authorities themselves had originally posted on the bulletin board; and a set of encrypted, mixed sums of credentials and ballots, posted on the bulletin board by the voters. The two lists have been encrypted with different Paillier public parameters. Using threshold protocols for the corresponding sets of private keys, the tallying authorities decrypt the elements in each list and then compare them through a search algorithm and publish the tallying result on bulletin board.

MODELING ACQUISTI PROTOCOL WITH APPLIED PI CALCULUS

Function and equational theory: The function and equational theory is introduced in this section. We use applied PI calculus to model Acquisti protocol. We model cryptography in a Dolev-Yao model as being perfect. Figure 14 describes the functions and Fig. 15 describes the equational theory in Acquisti protocol.

The probabilistic public key cryptosystem, for example Paillier cryptosystem, is modeled with decryption algorithm pPKdec(x,PR) and encryption algorithm pPKenc(x, PU,r). pPKdec(x, PR) decrypt the ciphertext x x with private key PR. pPKenc(x, PU, r) encrypt the plain text x with public key PU and random number r. The deterministic public key encryption scheme is expressed

Fun $\mathrm{pPKdec}(x,PR)$	Fun $\mathrm{add}(x,y)$
Fun $\mathrm{pPKenc}(x,PU,r)$	Fun $\mathrm{TpPKdec}(x_1,\cdots,x_2)$
Fun $\mathrm{PKdec}(x,PR)$	Fun $\mathrm{TPKdec}(x_1,\cdots,x_2)$
Fun $\mathrm{PK}(x)$	Fun $\mathrm{VK}(x)$
Fun $\mathrm{PKenc}(x,PU)$	Fun $\mathrm{verifysign}(x,PU)$
Fun $\mathrm{sign}(x,PR)$	Fun $\mathrm{projection}_i(x)$
Fun $\mathrm{decsign}(x,PU)$	Fun $\mathrm{SelfBlinding}(x,r)$
Fun $\mathrm{TpPKsubdec}(x_i,PR_i,VK_i)$	Fun $\mathrm{TPKsubdec}(x_i,PR_i,VK_i)$
Fun $\mathrm{checkciphertext}(x_1,x_2)$	Fun $\mathrm{SK}(x)$
Fun $\mathrm{equals}(x,y)$	

Fig. 14: Functions

$$\text{equation}\quad \mathrm{CheckNZDVPp}(\mathrm{DVPsign}(x,PR),VK_y,x)=\mathrm{true}.$$
$$\text{equation}\quad \mathrm{equals}(x,x)=\mathrm{true}.$$
$$\text{equation}\quad \mathrm{decsign}(\mathrm{sign}(x,PR))=x.$$
$$\text{equation}\quad \mathrm{verifysign}(\mathrm{sign}(x,PR),x)=\mathrm{true}.$$
$$\text{equation}\quad \mathrm{pPKdec}\big(\mathrm{SelfBlinding}(\mathrm{pPKenc}(x,PU_y,r),r_1),PR_y\big)=x.$$
$$\text{equation}\quad \mathrm{add}(\mathrm{projection}_1(x),\mathrm{projection}_2(x))=x.$$
$$\text{equation}\quad \mathrm{add}(\mathrm{projection}_2(x),\mathrm{projection}_1(x))=x.$$
$$\text{equation}\quad \mathrm{TpPKdec}\big(\mathrm{TpPKenc}(x,PU_y,r),PR_y\big)=x$$
$$\text{equation}\quad \mathrm{TpPKdec}\begin{pmatrix}\mathrm{TpPKsubdec}\big(\mathrm{TpPKenc}(x,PU_y,r_1),PR_1,VK_1\big),\\ \mathrm{TpPKsubdec}\big(\mathrm{TpPKenc}(x,PU_y,r_2),PR_2,VK_2\big)\end{pmatrix}=x$$
$$\text{equation}\quad \mathrm{TPKdec}\big(\mathrm{TPKenc}(x,PU_y),PR_y\big)=x$$
$$\text{equation}\quad \mathrm{TPKdec}\begin{pmatrix}\mathrm{TPKsubdec}\big(\mathrm{TPKenc}(x,PU_y),PR_1,VK_1\big),\\ \mathrm{TPKsubdec}\big(\mathrm{TpPKenc}(x,PU_y),PR_2,VK_2\big)\end{pmatrix}=x$$

Fig. 15: Equational theory

by decryption algorithm PKdnc(x,PR) and encryption algorithm Pkenc(x,PU). PKdnc(x,PR) decrypt the ciphertext x with private key PR. PKenc(x,PU) encrypt the plaintext x with public key PU. The digital signature is modeled as being signature with message recovery, i.e., the signature itself contains the signed message which can be extracted using the function. The digital signature algorithm includes the generation signature algorithm sign(x, PR) sign the message x with private key PR and the verification algorithm verifysign(x, PU) verify the digital signature x with public key PU. decsign(x, PU) recover the message from the digital signature x with public key PU. The probabilistic threshold public key share decryption algorithm TpPKsubdec(x_i, Pr_i, VK_i) decrypt the secret share x_i with private key PR_i and verification key VK_i. The probabilistic threshold combining algorithm TpPKdec(x_1,...,x_2) recovers x from x_1,...,x_2. The deterministic threshold public key share decryption algorithm TPKsubdec(x_i, PR_i, VK_i) decrypt the secret share x_i with private key PR_i and verification key VK_i. The deterministic threshold combining algorithm TPKdec(x_1,...,x_2) means that recover x from x_1,...,x_2. The projection function projection$_i$(x) generated the ith share from the formatted message x. The self blinding function SelfBlinding(x,r) blinds message x with r. The add function add(x,y) add x and y. checkciphertext(x_1,x_2) verify the two ciphertext x_1 and x_2 generated with the same plaintext. SK(x) PK(x) VK(x) generated the secret key, public key and verification key of x. equals(x,y) checks whether x is equal to y or not.

The basic equational theory is described in Fig. 15. The threshold decryption and combining algorithm are introduced.

The equational theory also contains and equational rules for abstractly reasoning about the knowledge proof that two ciphertexts are encryption of the same plaintext which is modeled in Fig. 16 and used in the voting phase and tallying phase. In the voting phase the voter need to verify the equivalence between the encrypted share and the one the voter has received to its message are also provided to itself. In the tallying phase the tally authority need to check the two lists: a list of encrypted, mixed credentials the registration authorities themselves had originally posted on the bulletin board; and a set of encrypted, mixed sums of credentials and ballots, posted on the bulletin board by the voters. It modeled as:

$$\text{checkciphertext}\left(\text{pPKenc}(x_1, PU_y, r_1), \text{pPKenc}(x_1, PU_z, r_2)\right) = \text{true}$$

It can verify the two ciphertext, one is the ciphertext generated with the public key PU_y and random number r_1, the other is the ciphertext generated with the public key PU_z and random number r_2, are the same plaintext x_1.

Designated verifier proofs modeled in Fig. 17 in the applied PI calculus (Backes *et al*., 2008b) is also contained in the equational theory and equational rules. During the registration phase, the voter acquires a private credential for voting. For this he contacts each of the registration authority and asks them for a share of the private credential. In order for a registration authority to prove the correctness of the share credential, a designated verifier proof is used. Designated verifier proof can convince only the voter of the correctness of the share credential and nobody else. In particular the voter should be able to generate a fake proof of this fact, e.g., using his secret key.

$$\text{CheckNZDVPp (DVPsing}(x,PR), VK_y, x) = \text{ture}$$

$$\boxed{\text{equation} \quad \text{checkciphertext}\left(\text{pPKenc}\left(x_1, PU_y, r_1\right), \text{pPKenc}\left(x_1, PU_z, r_2\right)\right) = \text{true.}}$$

Fig. 16: Model of knowledge proof that two ciphertexts are encryption of the same plaintext

$$\begin{array}{|l|}
\hline
\text{Public}_p(ZK_{i,j}(\widetilde{N},\widetilde{M}, F)) = N_p. \qquad p \in [1, j] \\
\text{Formula}(ZK_{i,j}(\widetilde{N},\widetilde{M}, F)) = F. \\
\text{Ver}_{i,j}(F, ZK_{i,j}(\widetilde{N},\widetilde{M}, F)) = \text{true} \quad \text{iff} \ \left[\ F\{\widetilde{N}/\widetilde{\alpha}\}\{\widetilde{M}/\widetilde{\beta}\} = \text{true} \right] \wedge \ [\ F \text{ is an } (i,j)-\text{formula}] \\
F' = F \vee \left[\beta_{j+1} = \text{check}(\alpha_{i+1}, \beta_{j+2}) \right] \\
ZK_{i+1,j+2}\left((N_1, \cdots, N_i, N_{i+1}), (M_1, \cdots, M_j), m, VK_y, F' \right) \\
\text{CheckNZDVPp}\left(\text{DVPsign}(m, PR), VK_y, m'\right) = \text{true} \quad \text{iff} \quad \text{equals}\left(\text{check}(\alpha_{i+1}, \beta_{j+2}), \beta_{j+1}\right) = \text{true} \\
\hline
\end{array}$$

Fig. 17: Model of designated verifier proofs

$$\begin{array}{l} \text{Acquisti protocol} \triangleq \text{new } C; \text{new } V; \text{new } S; \\ \quad \text{new keyV}; \text{new keyI}_1; \text{new keyI}_2; \\ \quad \text{new chVR}; \text{new chRI}_1; \text{new chRI}_2; \\ \quad \left(\begin{array}{l} !\text{voter} | !\text{corrupted voter} | !\text{tallying authority} | \\ !\text{issuer authority}_1 | !\text{issuer authority}_2 | !\text{registration authority} \end{array} \right) \end{array}$$

Fig. 18: Main process

$$\begin{array}{l} \text{voter} \triangleq \\ \quad \text{in(chVR,id);startid(id);} \\ \quad \text{in(chVR,kencNZDVP}_1\text{);} \\ \quad \text{in(chVR,kencNZDVP}_2\text{);} \\ \quad \text{let} \quad \text{NZDVP}_1 = \text{PKdec}[\text{kencNZDVP}_1,\text{SK(keyV)}] \quad \text{in} \\ \quad \text{let} \quad \text{NZDVP}_2 = \text{PKdec}[\text{kencNZDVP}_2,\text{SK(keyV)}] \quad \text{in} \\ \quad \text{if} \quad \text{CheckNZDVPp}\left(\text{DVPsign}\left(\text{Public}_3(\text{NZDVP}_1),\text{SK(keyV)}\right),\text{VK(keyV)},\text{Public}_3(\text{NZDVP}_1)\right) \quad \text{then} \\ \quad \text{if} \quad \text{CheckNZDVPp}\left(\text{DVPsign}\left(\text{Public}_3(\text{NZDVP}_2),\text{SK(keyV)}\right),\text{VK(keyV)},\text{Public}_3(\text{NZDVP}_2)\right) \quad \text{then} \\ \quad \text{if} \quad \text{checkciphertext}\left(\text{Public}_1(\text{NZDVP}_1),\text{decsign}\left(\text{Public}_2(\text{NZDVP}_1)\right)\right) = \text{true} \quad \text{then} \\ \quad \text{if} \quad \text{checkciphertext}\left(\text{Public}_1(\text{NZDVP}_2),\text{decsign}\left(\text{Public}_2(\text{NZDVP}_2)\right)\right) = \text{true} \quad \text{then} \\ \quad \text{let} \quad \text{cred} = \prod_{i=1,2} \text{Public}_1(\text{NZDVP}_i) \quad \text{in} \\ \quad \text{let} \quad \text{vote} = \prod_{i=1,2} \text{vencballot}_i^t \quad \text{in} \\ \quad \text{let} \quad \text{result} = \text{cred} \times \text{vote} \quad \text{in} \\ \quad \text{new} \quad \text{r;} \\ \quad \text{out}\left(\text{pub,TpPKenc(result,PK(S),r)}\right); \end{array}$$

Fig. 19: Voter process

$$\begin{array}{l} \text{corruptedvoter} \triangleq \\ \quad \text{in(chVR,id);startcorid(id);} \\ \quad \text{in(chVR,kencNZDVP}_1\text{);} \\ \quad \text{in(chVR,kencNZDVP}_2\text{);} \\ \quad \text{let} \quad \text{NZDVP}_1 = \text{PKdec}[\text{kencNZDVP}_1,\text{SK(keyV)}] \quad \text{in} \\ \quad \text{let} \quad \text{NZDVP}_2 = \text{PKdec}[\text{kencNZDVP}_2,\text{SK(keyV)}] \quad \text{in} \\ \quad \text{if} \quad \text{CheckNZDVPp}\left(\text{DVPsign}\left(\text{Public}_3(\text{NZDVP}_1),\text{SK(keyV)}\right),\text{VK(keyV)},\text{Public}_3(\text{NZDVP}_1)\right) \quad \text{then} \\ \quad \text{if} \quad \text{CheckNZDVPp}\left(\text{DVPsign}\left(\text{Public}_3(\text{NZDVP}_2),\text{SK(keyV)}\right),\text{VK(keyV)},\text{Public}_3(\text{NZDVP}_2)\right) \quad \text{then} \\ \quad \text{out}\left(\text{pub,}\left(\text{Public}_1(\text{NZDVP}_1),\text{Public}_1(\text{NZDVP}_2)\right)\right); \end{array}$$

Fig. 20: Corrupted voter process

Processes: The complete formal model of Acquisti protocol in applied PI calculus is given in Fig. 18-23 report the basic process include main process, voter process, corrupted voter process, registration authority process, issuer authority process and tallying authority process forming our of the model of Acquisti protocol. Figure 24-29 offer additional and modified processes for the analysis of coercion-resistance.

The main process in Fig. 18 sets up private channels chVR, $chRI_1$, $chRI_2$ and specifies how the processes are combined in parallel. chVR is the private channel between voter and registration authority. $chRI_1$ and $chRI_2$ are the private channel between registration authority and issuer authority. At the same time the main process generates the key parameters c for credentials, V for vote, S for non-homomorphic cryptosystem, keyV for voter and key1 for issuer authority.

$$
\begin{aligned}
&\text{registration authority} \triangleq \\
&\quad \text{new id; newid(id);} \\
&\quad \text{out(pub,id);} \\
&\quad \text{out(chVR,id);} \\
&\quad \text{new cred;} \\
&\quad \text{out}(\text{chRI}_1,(\text{id},\text{cred})); \\
&\quad \text{out}(\text{chRI}_2,(\text{id},\text{cred})); \\
&\quad \text{in}(\text{chRI}_1,(\text{id},\text{cred}_1)); \\
&\quad \text{in}(\text{chRI}_2,(\text{id},\text{cred}_2)); \\
&\quad \text{new } r_1; \text{new } r_2; \\
&\quad \text{out}\left(\text{chVR},\text{PKenc}\left(\text{NZDVP}_1,\text{PK}(\text{keyV})\right)\right); \\
&\quad \text{out}\left(\text{chVR},\text{PKenc}\left(\text{NZDVP}_2,\text{PK}(\text{keyV})\right)\right); \\
&\quad \text{NZDVP}_i = \text{ZK}_{6,4}\left(\begin{array}{l}\text{cred}_i,r_1,r_2,C,V,\text{DVPsign}(m,\text{SK}(\text{keyV}));\text{pPKenc}(\text{cred}_i,\text{PK}(V),r_1),\\ \text{sign}(\text{pPKenc}(\text{cred}_i,\text{PK}(C),r_2),\text{SK}_i(C)),m,\text{VK}(\text{keyV});\end{array}\right)
\end{aligned}
$$

Fig. 21: Registration authority process

$$
\begin{aligned}
&\text{issuer authority}_{i\in\{1,2\}} \triangleq \\
&\quad \text{new } r_1; \text{new } r_2; \\
&\quad \text{out}\left(\text{pub},\left(\text{sign}\left\{\left[\begin{array}{l}\text{pPKenc}\left(\text{projection}_i(\text{ballot}^t),\text{PK}(C),r_1\right),\\ \text{pPKenc}\left(\text{projection}_i(\text{ballot}^t),\text{PK}(V),r_2\right)\end{array}\right],\text{SK}\left(\text{keyI}_i\right)\right\},\text{zkpe}\right)\right). \\
&\quad \text{in}(\text{chRI}_i,(\text{id},\text{cred})); \\
&\quad \text{out}(\text{chRI}_i,(\text{id},c_i(\text{cred}))); \\
&\quad \text{out}\left(\text{pub},\text{sign}\left[\text{pPKenc}(c_i(\text{cred}),\text{PK}(C),r_1),\text{SK}_i(C)\right]\right);
\end{aligned}
$$

Fig. 22: Issuer authority process

$$
\begin{aligned}
&\text{tallying authority} \triangleq \\
&\quad \text{let} \quad \text{cenccred} = \prod_{i=1,2}\text{TpPKdec}(\text{projection}_i(\text{cred}),\text{PK}(C),r) \quad \text{in} \\
&\quad \text{let} \quad \text{bcenccred} = \text{SelfBlinding}(\text{cenccred},\text{PK}(C)) \quad \text{in} \\
&\quad \text{in(pub,res);} \\
&\quad \text{let} \quad \text{venccredvote} = \text{TpPKdec}(\text{res},\text{SK}(S)) \quad \text{in} \\
&\quad \text{let} \quad \text{bvenccredvote} = \text{SelfBlinding}(\text{venccredvote},\text{PK}(V)) \quad \text{in} \\
&\quad \text{let} \quad \text{cencballot}^t = \prod_{i=1,2}\text{TpPKenc}(\text{projection}_i(\text{ballot}^t),\text{PK}(C),r) \quad \text{in} \\
&\quad \text{let} \quad \text{test} = \text{bcenccred}\times\text{cencballot}^t \quad \text{in} \\
&\quad \text{if} \quad \text{true} = \text{checkciphertext}(\text{test},\text{bvenccredvote}) \quad \text{then} \\
&\quad \text{endvote}(\text{ballot}^t);
\end{aligned}
$$

Fig. 23: Tallying authority process

Voter process is modeled in applied PI calculus in Fig. 19. Using Paillier encryption, each voter get the shares ciphertext kVenccred_1 and kVenccred $_2$from registration authority, then decrypt and get the credentials venccred_1, venccred_2 and the designated verifier proof NZDVP_1 and NZDVP_2. After that the voter use CheckNZDVPp,to verify NZDVP_1 and NZDVP_2. The voter also use checkciphertest( ) to verify the equivalence between the encrypted share: Public_1 (NZDVP_1), decsign (Public_2 (NZCVP_1)) and the one Public_1 (NZDVP_2), decsign (Public_2 (NZCVP_2)) the voter has received to its message are also provided to itself. The voter also gets the encrypted shares vencballot^t_t of the ballot, which she has selected from the bulletin board. He multiplies:

$$
\begin{array}{l}
\text{cheating voter} \triangleq \\
\quad \text{in}(\text{chVR}, \text{id}); \\
\quad \text{in}(\text{chVR}, \text{kencNZDVP}_1); \\
\quad \text{in}(\text{chVR}, \text{kencNZDVP}_2); \\
\quad \text{let} \quad \text{NZDVP}_1 = \text{PKdec}(\text{kencNZDVP}_1, \text{SK}(\text{keyV})) \quad \text{in} \\
\quad \text{let} \quad \text{NZDVP}_2 = \text{PKdec}(\text{kencNZDVP}_2, \text{SK}(\text{keyV})) \quad \text{in} \\
\quad \text{if} \quad \text{CheckNZDVPp}(\text{DVPsign}(\text{Public}_3(\text{NZDVP}_1), \text{SK}(\text{keyV})), \text{VK}(\text{keyV}), \text{Public}_3(\text{NZDVP}_1)) \quad \text{then} \\
\quad \text{if} \quad \text{CheckNZDVPp}(\text{DVPsign}(\text{Public}_3(\text{NZDVP}_2), \text{SK}(\text{keyV})), \text{VK}(\text{keyV}), \text{Public}_3(\text{NZDVP}_2)) \quad \text{then} \\
\quad \text{if} \quad \text{checkciphertext}(\text{Public}_1(\text{NZDVP}_1), \text{decsign}(\text{Public}_2(\text{NZDVP}_1))) = \text{true} \quad \text{then} \\
\quad \text{if} \quad \text{checkciphertext}(\text{Public}_1(\text{NZDVP}_2), \text{decsign}(\text{Public}_2(\text{NZDVP}_2))) = \text{true} \quad \text{then} \\
\quad \text{new} \ \text{fakecred}_1; \text{new} \quad \text{fakecred}_2; \\
\quad \text{out}(\text{c}, (\text{fakecred}_1, \text{fakecred}_2)); \\
\quad \text{out}(\text{chve}, (\text{fakecred}_1, \text{fakecred}_2)); | \\
\quad \text{let} \quad \text{cred} = \prod_{i=1,2} \text{Public}_1(\text{NZDVP}_i) \quad \text{in} \\
\quad \text{let} \quad \text{vote} = \prod_{i=1,2} \text{vencballot}_i^t \quad \text{in} \\
\quad \text{let} \quad \text{result} = \text{cred} \times \text{vote} \quad \text{in} \\
\quad \text{new} \quad \text{r}; \\
\quad \text{out}(\text{pub}, \text{TpPKenc}(\text{result}, \text{PK}(\text{S}), \text{r}));
\end{array}
$$

Fig. 24: Cheating voter process

$$
\begin{array}{l}
\text{coerced voter} \triangleq \\
\quad \text{in}(\text{chVR}, \text{id}); \\
\quad \text{in}(\text{chVR}, \text{kencNZDVP}_1); \\
\quad \text{in}(\text{chVR}, \text{kencNZDVP}_2); \\
\quad \text{let} \quad \text{NZDVP}_1 = \text{PKdec}(\text{kencNZDVP}_1, \text{SK}(\text{keyV})) \quad \text{in} \\
\quad \text{let} \quad \text{NZDVP}_2 = \text{PKdec}(\text{kencNZDVP}_2, \text{SK}(\text{keyV})) \quad \text{in} \\
\quad \text{if} \quad \text{CheckNZDVPp}(\text{DVPsign}(\text{Public}_3(\text{NZDVP}_1), \text{SK}(\text{keyV})), \text{VK}(\text{keyV}), \text{Public}_3(\text{NZDVP}_1)) \quad \text{then} \\
\quad \text{if} \quad \text{CheckNZDVPp}(\text{DVPsign}(\text{Public}_3(\text{NZDVP}_2), \text{SK}(\text{keyV})), \text{VK}(\text{keyV}), \text{Public}_3(\text{NZDVP}_2)) \quad \text{then} \\
\quad \text{if} \quad \text{checkciphertext}(\text{Public}_1(\text{NZDVP}_1), \text{decsign}(\text{Public}_2(\text{NZDVP}_1))) = \text{true} \quad \text{then} \\
\quad \text{if} \quad \text{checkciphertext}(\text{Public}_1(\text{NZDVP}_2), \text{decsign}(\text{Public}_2(\text{NZDVP}_2))) = \text{true} \quad \text{then} \\
\quad \text{out}(\text{c}, (\text{Public}_1(\text{NZDVP}_1), \text{Public}_1(\text{NZDVP}_2))); \\
\quad \text{out}(\text{chve}, (\text{Public}_1(\text{NZDVP}_1), \text{Public}_1(\text{NZDVP}_2))).
\end{array}
$$

Fig. 25: Coerced voter process

$$\text{cred} = \prod_{i=1,2} \text{venccred}_i \quad \text{and} \quad \text{vote} = \prod_{i=1,2} \text{vencballot}_i^t$$

Because of the homomorphic properties of Paillier cryptosystems, the resulting ciphertext result includes the sum of credential shares and the ballot's shares. The resulting ciphertext TpPKenc(result, PK(S), r) is sent to the bulletin board.

Corrupted voters process is modeled in Fig. 20. The corrupted voter will register and get his secret credentials shares kVenccred_1 and kVenccred from registration authority, then decrypt and get the credentials venccred_1, venccred_2 proof of the equivalence between the encrypted share it has posted on the bulletin board and the one it will sent to the voter. NZDVP_1 and NZDVP_2, after that, he simply output all their registration secrets venccred_1 and

$$
\begin{array}{l}
\text{modified tallying authority} \triangleq \\
\quad \text{let} \quad \text{cenccred} = \prod_{i=1,2} \text{TpPKdec}(\text{projection}_i(\text{cred}), \text{PK}(C), r) \quad \text{in} \\
\quad \text{let} \quad \text{bcenccred} = \text{SelfBlinding}(\text{cenccred}, \text{PK}(C)) \quad \text{in} \\
\quad \text{in}(\text{pub}, \text{res}); \\
\quad \text{let} \quad \text{venccredvote} = \text{TpPKdec}(\text{res}, \text{SK}(S)) \quad \text{in} \\
\quad \text{out}(\text{chTE}, \text{venccredvote}); \\
\quad \text{let} \quad \text{bvenccredvote} = \text{SelfBlinding}(\text{venccredvote}, \text{PK}(V)) \quad \text{in} \\
\quad \text{let} \quad \text{cencballot}^t = \prod_{i=1,2} \text{TpPKenc}(\text{projection}_i(\text{ballot}^t), \text{PK}(C), r) \quad \text{in} \\
\quad \text{let} \quad \text{test} = \text{bcenccred} \times \text{cencballot}^t \quad \text{in} \\
\quad \text{if} \quad \text{true} = \text{checkciphertext}(\text{test}, \text{bvenccredvote}) \quad \text{then}
\end{array}
$$

Fig. 26: Modified tallying authority process

$$
\begin{array}{l}
\text{abstained voter} \triangleq \\
\quad \text{in}(\text{chVR}, \text{id}); \\
\quad \text{in}(\text{chVR}, \text{kencNZDVP}_1); \\
\quad \text{in}(\text{chVR}, \text{kencNZDVP}_2);
\end{array}
$$

Fig. 27: Abstained voter process

$$
\begin{array}{l}
\text{extractor}[\,] \triangleq \\
\quad \text{in}(\text{chVR}, \text{id}); \\
\quad \text{in}(\text{chVR}, \text{kencNZDVP}_1); \\
\quad \text{in}(\text{chVR}, \text{kencNZDVP}_2); \\
\quad \text{new} \quad a; \text{new} \quad b; \\
\quad \left(\text{in}\left(\text{chve}, \left(\text{fakecred}_1, \text{fakecred}_2\right)\right); !\text{out}\left(a, \left(\text{fakecred}_1, \text{fakecred}_2\right)\right)\right) | \\
\quad (!\text{in}(\text{chTE}, \text{venccredvote}); \\
\quad \text{in}(a, (\text{fakecred}_1, \text{fakecred}_2)); \\
\quad \text{let} \quad \text{vote}^t = \prod_{i=1,2} \text{vencballot}_i^t \quad \text{in} \\
\quad \text{let} \quad \text{cred} = \prod_{i=1,2} \text{fakecred}_i \quad \text{in} \\
\quad \text{if} \quad \text{venccredvote} = \text{vote} \times \text{cred} \quad \text{then} \\
\quad \text{out}(b, \text{vote}^t)) \,| \\
\quad (\text{in}(b, z); \\
\quad \text{if} \quad z \in \text{ballot}^t \quad \text{then} \quad [\,])
\end{array}
$$

Fig. 28: Extractor process

venccred_2 on a public channel, so that the attacker can impersonate them in order to mount any sort of attack.

The registration authority process is modeled in Fig. 21. The registration authority generate the voters id, then get the secret credentials shares cred_1 and cred_2 After that the registration authority creates designated verifier proof that the proof of the equivalence between the encrypted share it has posted on the bulletin board and the one it will sent to the voter NZDVP_1 and NZDVP_1.

The issuer authority is modeled in Fig. 22. The issuer authorities get the shares of ballot by $\text{projection}_i(\text{ballot}^t)$ and send sign[pPKenc(c_i(cred, PK(C), r_1, $SK_i(C)$)] which is encrypted with a set of Paillier public parameters by the public channel pub.

$$\begin{array}{l}\text{Acquisti} - \text{coercion} - \text{resistance1} \triangleq \text{new } C; \text{new } V; \text{new } S;\\ \quad \text{new keyV}; \text{new keyI}_1; \text{new keyI}_2;\\ \quad \text{new chVR}; \text{new chRI}_1; \text{new chRI}_2; \text{new chVE}; \text{new chTE};\\ \quad \left(\begin{array}{l}!\text{voter} \mid !\text{corrupted voter} \mid !\text{modified tallying authority} \mid\\ !\text{issuer authority}_1 \mid !\text{issuer authority}_2 \mid !\text{registration authority} \mid\\ \text{coerced voter} \mid \text{voter}(v_a) \mid \text{extractor}(0)\end{array}\right)\end{array}$$

Fig. 29: Acquisti-coercion-resistance2 process

$$\begin{array}{l}\text{Acquisti} - \text{coercion} - \text{resistauce2} \triangleq \text{new } C; \text{new } V; \text{new } S;\\ \quad \text{new keyV}; \text{new keyI}_1; \text{new keyI}_2;\\ \quad \text{new chVR}; \text{new chRI}_1; \text{new chRI}_2; \text{new chVE}; \text{new chTE};\\ \quad \left(\begin{array}{l}!\text{voter} \mid !\text{corrupted voter} \mid !\text{ modified tallying authority} \mid\\ !\text{issuer authority}_1 \mid !\text{issuer authority}_2 \mid !\text{registration authority} \mid\\ \text{cheating voter} \mid \text{abstained voter} \mid \text{extractor}(\text{out}(\text{chvote}, z))\end{array}\right)\end{array}$$

Fig. 30: Acquisti-coercion-resistance2 process

Tallying authority process is modeled in Fig. 23. After the voting time expires, the tallying authorities get the all ballots on bulletin board posted by allegedly eligible voters and then mixed it by SelfBlinding(cenccred, PK(C)). The shares of credentials posted by the registration authorities are also combined and then mixed SelfBlinding(venccredvote, PK(V)). Tallying authorities thus obtain two lists: a list bcenccred of encrypted, mixed credentials the registration authorities themselves had originally posted on the bulletin board; and a set cencballot^t of encrypted, mixed sums of credentials and ballots, posted on the bulletin board by the voters. The two lists have been encrypted with different Paillier public parameters. Using threshold protocols for the corresponding sets of private keys, the tallying authorities decrypt the elements in each list by checkciphertext(test, bvenccredvote) and then compare them through a search algorithm and publish the tallying result on bulletin board.

According to the definition coerced-resistance of Backes *et al.* (2008a) in order to analyze coercion-resistance of Acquisti protocol, the processes including cheating voter process, coerced voter process, modified tallying authority process, abstained voter process, extractor process, Acquisti-coercion-resistance1 process and Acquisti-coercion-resistance2 process, are needed. The faking strategy of the cheating voter consists of generating a fake credential and sending it to the coercer. To generate the fake credential fakecred_1 and fakecred_2, the voter construct a valid designated verifier proof NZDVP that causes this fake share to appear real to the coercer in Fig. 24. In Fig. 25 the coerced voter sends his genuine (Public_1 (NZDVP_1), Public_1 (NZDVP_2)) and (Public_1 (NZDVP_1), Public_1 (NZDVP_2)) to the coercer.

In Fig. 26 the modified tallying authority sends the credentials and vote by chTE to the extractor process. The difference between the previous tallying authority process in Fig. 23 and modified tallying authority process is that the tallying authority process in Fig. 23 does not publish the credentials and vote. In Fig. 27 the abstained voter receives the related information and give up his vote. The extractor process in Fig. 28 can identify this fake designated verifier proof as being a coerced vote. Notice, that the modified tallying authority process in Fig. 26 shares a private channel chTE with the extractor aprivate channel chVE with voter. The processes Acquisti-coercion-resistance1 in Fig. 29 and Acquisti-coercion-resistance2 in Fig. 30 need to be observationally equivalent in order to satisfy the definition coercion resistance of Backes *et al.* (2008a) and to be able to mechanizedly prove this property of the protocol.

MECHANIZED PROOF OF ACQUISTI PROTOCOL WITH PROVERIF

ProVerif can take two formats as input. The first one is in the form of Horn clauses (logic programming rules) and applied PI calculus. The second one is in the form of a process in an extension of the PI calculus (Abadi and Blanchet, 2005). In both cases, the output of the system is essentially the same.

In this study we use an extension of the PI calculus as the input of ProVerif. In order to prove the soundness and coercion resistance in Acquisti protocol the applied PI calculus model are needed to be translated into the syntax of ProVerif and generated the ProVerif inputs in extension of the PI calculus (Abadi and Blanchet, 2005).

```
fun    pPKenc/3.  (* probabilistic public key encryption *)
fun    pPKdec/2.(* probabilistic public key decryption *)
fun    PKenc/2.(* deterministic public key encryption *)
fun    PKdec/2.(* deterministic public key decryption *)
fun    check/2.
fun    zkver/1.
fun    public1/1.
fun    zk/2.
data   true/0.
fun    sign/2.(* digital signature algorithm with the private key *)
(*recover the message from the digital signature with public key *)
fun    decsign/2.
fun    verifysign/2.(*verify the digital signature  with public key *)
(*deterministic threshold public key share decryption algorithm *)
fun    TPKsubdec/3.
(*probabilistic threshold public key share decryption algorithm *)
fun    TpPKsubdec/3.
fun    TPKdec/2.(*deterministic threshold combining algorithm *)
fun    TPKenc/2.
fun    TpPKenc/3.
fun    TpPKdec/2.(*probabilistic threshold combining algorithm *)
```

Fig. 31: Function 1

```
fun    SK/1.(* algorithm of generating the private key *)
fun    VK/1.(* algorithm of generating the verification key *)
fun    PK/1.(*algorithm of generating the public key *)
(* verification of knowledge proof that two
 ciphertexts are encryption of the same plaintext *)
fun    checkciphertext/4.
fun    add/2.(* add *)
fun    multi/2.(*multiply *)
fun    equals/2.(*equation *)
fun    selfBlinding/2.(*self blinding *)
fun    projection1/1.(*projection function *)
fun    projection2/1.(*projection function *)
```

Fig. 32: Function 2

Firstly the soundness of Acquisti protocol is proved by ProVerif. In order to prove the soundness property, according to the definition of Backes *et al*. (2008a) model, the soundness consists of inalterability(condition 1a), ligibility(condition 1b) and non-reusability(condition 1a and condition 2). Figure 31-40 give the inputs in extension of the PI calculus (Abadi and Blanchet, 2005) of verification of soundness in ProVerif. The analysis was performed by ProVerif and succeeded in Fig. 41. Bad is not derivable shows that observational equivalence is true. As a result, Acquisti protocol is proved to guarantee inalterability, eligibility and unbreusability for an unbounded number of honest voters and an unbounded number of corrupted participants.

The proof of coercion-resistance in Acquisti protocol is also finished by ProVerif. According to definition of coercion-resistance in Backes *et al*. (2008a) model, the coercion-resistance is composed of one hypothesis and four conditions. The hypothesis describes that election context S' that only differs from S in that the tallying authority additionally outputs messages on the channel c_2 shared with Extractor.

```
equation equals(x,x)=true.
equation pPKdec(pPKenc(x,PK(y),z),SK(y))=x.
equation PKdec(PKenc(x,PK(y),SK(y))=x.
equation decsign(sign(x,SK(y)),PK(y))=x.
equation verifysign(sign(x,SK(y)),x)=true.
equation check(sign(x,y),VK(y))=x.
equation add(projection1(x),projection2(x))=x.
equation add(projection2(x),projection1(x))=x.
equation TPKdec(TPKenc(x,PK(y)),SK(y))=x.
equation public1(zk(x,y))=y.
equation multi(TpPKenc(a,PK(y),r),TpPKenc(b,PK(y),z))=TpPKenc(add(a,b),PK(y),r)).
equation checkciphertext(TpPKenc(x,PK(y),r1),TpPKenc(x,PK(z),r2),y,z)=true.
equation TPKdec(TPKsubdec(TPKenc(x,PK(y)),projection1(SK(y)),projection1(VK(y))),
                TPKsubdec(TPKenc(x,PK(y)),projection2(SK(y)),projection2(VK(y))))=x.
equation TPKdec(TPKsubdec(TPKenc(x,PK(y)),projection2(SK(y)),projection2(VK(y))),
                TPKsubdec(TPKenc(x,PK(y)),projection1(SK(y)),projection1(VK(y))))=x.
equation TpPKdec(TpPKsubdec(TpPKenc(x,PK(y),r1),projection1(SK(y)),projection1(VK(y))),
                 TpPKsubdec(TpPKenc(x,PK(y),r2),projection2(SK(y)),projection2(VK(y))))=x.
equation TpPKdec(TpPKsubdec(TpPKenc(x,PK(y),r2),projection2(SK(y)),projection2(VK(y))),
                 TpPKsubdec(TpPKenc(x,PK(y),r1),projection1(SK(y)),projection1(VK(y))))=x.
equation zkver(zk((cred1,cred2,sign(m,voter)),(TpPKenc(cred1,PK(V),r1),
                 TpPKenc(cred2,PK(V),r2),PK(V),m,VK(voter))))=true.
```

Fig. 33: Equation

```
(*public channel *)
free pub.
private free chvote.
free va,vb.
free n1,n2.
query evinj:ENDVOTE(x) ((evinj:BEGINVOTE(x,y)
                         evinj: STARTID(y)) | evinj:STARTCORID(z)).
(*voter *)
let votechooser =  out(chvote,va) | out(chvote,vb).
```

Fig. 34: Soundness-vote chooser

Condition 2 describes the special observational equivalence between:

$$S'\left[V_i^{coerced(c,c_1)} \middle| V_j(v') \middle| E_k^{c_1,c_2,z}[0]\right]$$

and

$$S'\left[V_i^{cheat(c,c_1)}(v') \middle| V_j^{abs} \middle| E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle z\rangle\right]\right]$$

$$S'\left[V_i^{coerced(c,c_1)} \middle| V_j(v') \middle| E_k^{c_1,c_2,z}[0]\right]$$

contains the voter V_i that is in accordance with the orders of the coercer, running in parallel with the voter V_j casting

```
let voter=
    new nonce;
    new nonce1;
    out(chVR,(n1,nonce));
    out(chVII,(n1,nonce1));
    in(chVII,(=n2,=nonce1,id));event STARTID(id);
    in(chVR,(=n2,=nonce,ct));
    let zkp=PKdec(ct,SK(voter)) in
    if zkver(zkp)=true then
    let (enccred1,enccred2,=PK(V),m,vk)=public1(zkp) in
     if check(sign(m,voter),vk)=m then
    let cred=multi(enccred1,enccred2) in
    in(chvote,vote);event BEGINVOTE(vote,id);
    new r1;new r2;
    let encvote=multi( TpPKenc(projection1(vote),PK(V),r1),
                       TpPKenc(projection2((vote),PK(V),r2)) ) in
    let ballot=multi(cred,encvote) in
    let res=PKenc(ballot,PK(S)) in
    out(pub,res).
```

Fig. 35: Soundness-voter

```
let corruptedvoter=
    new nonce;
    new nonce1;
    out(chVR,(n1,nonce));
    out(chVII,(n1,nonce1));
    in(chVII,(=n2,=nonce1,id));event STARTCORID(id);
    in(chVR,(=n2,=nonce,ct1));
    out(pub,ct1).
```

Fig. 36: Soundness-corrupted voter

a vote v' and the process $E_k^{c_1,c_2,z}[0]$, that is intuitively equivalent to a voter nullifying her vote. In:

$$S'\left[V_i^{cheat(c,c_1)}(v')\middle|V_j^{abs}\middle|E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle z\rangle\right]\right]$$

the voter V_i cheats the coercer by providing him with fake registration secrets and then votes v', the voter V_i participates in the registration phase and then abstains and the extractor process:

$$E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle z\rangle\right]$$

tallies the vote the coercer casts on behalf of V_i. Figure 31-33, 42-55 give the inputs in extension of the PI calculus (Abadi and Blanchet, 2005) of verification of Condition 2 in ProVerif. The result shows that the observational equivalence between:

$$S'\left[V_i^{coerced(c,c_1)}\middle|V_j(v')\middle|E_k^{c_1,c_2,z}[0]\right]$$

and

$$S'\left[V_i^{cheat(c,c_1)}(v')\middle|V_j^{abs}\middle|E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle z\rangle\right]\right]$$

is satisfied in Fig. 56. Bad is not derivable shows that observational equivalence is true.

```
let tallying_authority=
    new nonce;
    out(chRT,(n1,nonce));
    in(chRT,(=n2,=nonce,enccred1,enccred2));
    in(pub,res);
    let cenccred=multi[enccred1,enccred2] in
    let result=PKdec(res,SK(S)) in
    new r1;new r2;
    let cencvotea=multi(TpPKenc(projection1(va),PK(C),r1),
                        TpPKenc(projection2(va),PK(C),r2)) in
    let cencvoteb=multi(TpPKenc(projection1(va),PK(C),r1),
                        TpPKenc(projection2(vb),PK(C),r2)) in
    let test1=multi(cenccred,cencvotea) in
    let test2=multi(cenccred,cencvoteb) in
    if true=checkciphertext(test1,result,C,V) then event ENDVOTE(va) else
    if true=checkciphertext(test2,result,C,V) then event ENDVOTE(vb).
```

Fig. 37: Soundness-tallying authority

```
let registration_authority=
    in(chVR,(=n1,nonceV));
    in(chRT,(=n1,nonceT));
    new nonce;
    out(chIIR,(n1,nonce));
    in(chIIR,(=n2,=nonce,id));
    new cred;
    let cred1=projection1(cred) in
    let cred2=projection2(cred) in
    new r1;new r2;
    out(chRT,(n2,nonceT,TpPKenc(cred1,PK(C),r1),TpPKenc(cred2,PK(C),r2)));
    new m;new r3;new r4;
    out[chVR,(n2,nonceV,PKenc(zk((cred1,cred2,sign(m,voter)),
                                 (TpPKenc(cred1,PK(V),r3),
                                  TpPKenc(cred2,PK(V),r4),
                                  PK(V),m,VK(voter))),PK(voter)))].
```

Fig. 38: Soundness-registration authority

```
let issuer_authority=
    in(chVII,(=n1,nonceV));
    in(chIIR,(=n1,nonceR));
    new id; event NEWID(id);
    out(chVII,(n2,nonceV,id));
    out(chIIR,(n2,nonceR,id));
    out(pub,id).
```

Fig. 39: Soundness-issuer authority

```
process new C;new V;new S;
    new voter;
    new A1;new A2;
    new chVR;
    new chVII;
    new chIIR;
    new chRT;
    out(pub,PK(C));
    out(pub,PK(V));
    out(pub,PK(S));
    out(pub,PK(voter));
    out(pub,PK(A1));
    out(pub,PK(A2));
    (!voter|!corruptedvoter|!tallying_authority|!registration_authority
    |!issuer_authority|!votechooser)
```

Fig. 40: Soundness process

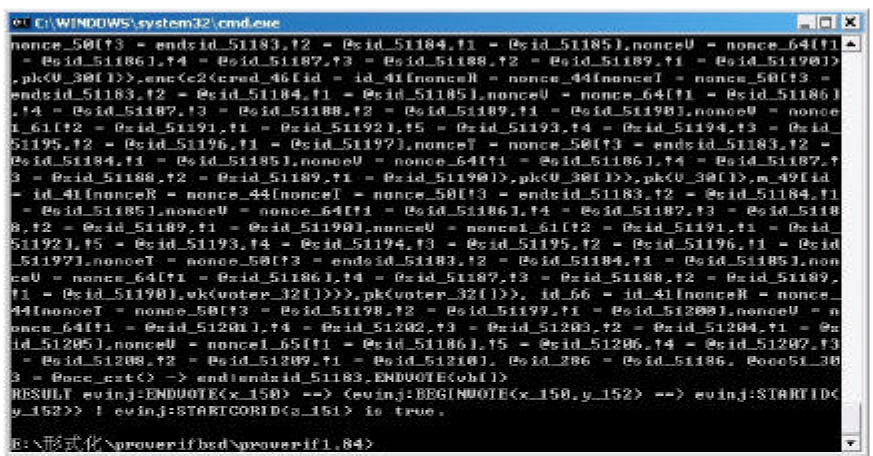


Fig. 41: The result of soundness

$$\text{equation}\ \ \text{zkver}\left(\text{zk}\left(\begin{array}{l}(\text{cred},\text{sign}(\text{m},\text{voter})),\\ \left(\begin{array}{l}(\text{TpPKenc}(\text{cred},\text{PK}(\text{V}),\text{r}),\\ \text{PK}(\text{V}),\text{m},\text{VK}(\text{voter})\end{array}\right)\end{array}\right)\right)=\text{true}.$$

Fig. 42: Coercion-resistance-condition2-additional equation

```
free pub,com.(*public channel*)
private free c,chvote,chTE,chVE,internal1,internal.
free va,vb.
free n1,n2.
(*voter*)
let votechooser = out(chvote,va)|out(chvote,vb)
```

Fig. 43: Coercion-resistance-condition2-vote chooser

```
let voter=
    in(chVII,id);
    new nonce;
    out(chVR,(n1,nonce));
    in(chVR,(=n2,=nonce,ct));
    let zkp=PKdec(ct,SK(voter)) in
    if zkver(zkp)=true then
    let (enccred1,enccred2,=PK(V),m,vk)=public1(zkp) in
    if check(sign(m,voter),vk)=m then
    let cred=multi(enccred1,enccred2) in
    in(chvote,vote);
    new r1;new r2;
    let encvote=multi(TpPKenc(projection1(vote),PK(V),r1),
                      TpPKenc(projection2(vote),PK(V),r2)) in
    let ballot=multi(cred,encvote) in
    let res=PKenc(ballot,PK(S)) in
    out(pub,res).
```

Fig. 44: Coercion-resistance-condition2-voter

```
let corruptedvoter=
    in(chVII,id);
    new nonce;
    out(chVR,(n1,nonce));
    in(chVR,(=n2,=nonce,ct));
    out(pub,ct).
```

Fig. 45: Coercion-resistance-condition2-corrupted voter

```
let voterreg =
    new nonce1;
    out(chVR2,(n1,nonce1));
    in(chVR2,(=n2,=nonce1,credI,credJ,credE));
    out(internal1,(n2,credI,credJ,credE)).
```

Fig. 46: Coercion-resistance-condition2-voter registration

Condition 3 describes the scenario that if the cheated coercer abstains, then the Extractor needs to abstain as well. The observational equivalence between:

$$vc.S'\left[!c(x)\middle|V_i^{cheat(c,c_1)}(v')\middle|V_j^{abs}\middle|E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle z\rangle\right]\right]$$

and

$$S\left[V_i(v')\middle|V_j^{abs}\middle|V_k^{abs}\right]$$

```
let coercedvoter =
    in(chVII,id);
    new nonce;
    out(chVR1,(n1,nonce));
    in(chVR1,(=n2,=nonce,ct));
    let zkp=PKdec(ct,SK(voter)) in
    if zkver(zkp)=true then
    let (enccred,=PK(V),m,vk)=public1(zkp) in
    if check(sign(m,voter),vk)=m then
    let cred=enccred in
    new fakecred;
    out(c,choice[cred,fakecred]);
    out(internal,(n2,cred,fakecred)).
```

Fig. 47: Coercion-resistance-condition2-coerced voter

```
let votercast =
    in(internal,(=n2,credI,fakecred));
    in(internal1,(=n2,=credI,credJ,credE));
    out(chVE,choice[(credI,credI,credJ,credE),(fakecred,credI,credJ,credE)]);
    new r;
    let encvote=TPKenc(va,PK(V)) in
    let res=PKenc((choice[credJ,credI],encvote),PK(S)) in
    out(pub,res).
```

Fig. 48: Coercion-resistance-condition2-voter cast

```
let tallying_authority=
  new nonce;
    out(chRT,(n1,nonce));
  in(chRT,(=n2,=nonce,enccred1,enccred2));
  in(pub,res);
  let cenccred=multi(enccred1,enccred2) in
  let result=PKdec(res,SK(S)) in
  new r1;new r2;
  let cencvotea=multi(TpPKenc(projection1(va),PK(C),r1),
                      TpPKenc(projection2(va),PK(C),r2)) in
  let cencvoteb=multi(TpPKenc(projection1(vb),PK(C),r1),
                      TpPKenc(projection2(vb),PK(C),r2)) in
  let test1=multi(cenccred,cencvotea) in
  let test2=multi(cenccred,cencvoteb) in
  if true=checkciphertext(test1,result,C,V) then out(com,va) else
    if true=checkciphertext(test2,result,C,V) then out(com,vb).
```

Fig. 49: Coercion-resistance-condition2-tallying authority

```
let registration_authority=
  in(chIIR,id);
    in(chVR,(=n1,nonceV));
  in(chRT,(=n1,nonceT));
    new cred;
  let cred1=projection1(cred) in
    let cred2=projection2(cred) in
    new r1;new r2;
  out(chRT,(n2,nonceT,TpPKenc(cred1,PK(C),r1),TpPKenc(cred2,PK(C),r2)));
    new m;new r3;new r4;
    out[chVR,(n2,nonceV,PKenc(zk((cred1,cred2,sign(m,voter)),
                                 (TpPKenc(cred1,PK(V),r3),
                                  TpPKenc(cred2,PK(V),r4),
                                  PK(V),m,VK(voter))),PK(voter)))].
```

Fig. 50: Coercion-resistance-condition2-registration authority

```
let issuer_authority=
  new id;
  out(chVII,id);
  out(chIIR,id);
  out(pub,id).
```

Fig. 51: Coercion-resistance-condition2-issuer authority

is need to be proved. Figure 31-33, 57-69 give the inputs in extension of the PI calculus (Abadi and Blanchet, 2005) of verification of Condition 3 in ProVerif. The result shows that the observational equivalence between:

$$vc.S'\left[!c(x)\middle|V_i^{cheat(c,c_1)}(v')\middle|V_j^{abs}\middle|E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle z\rangle\right]\right]$$

and

$$S\left[V_i(v')\middle|V_j^{abs}\middle|V_k^{abs}\right]$$

is true in Fig. 70. Bad is not derivable shows that observational equivalence is true.

In condition 4 if the cheated coercer casts a valid vote using the fake registration secrets he received from V_i, the Extractor needs to tally precisely this vote. The observational equivalence between:

$$vc.S'\left[P \,\middle|\, V_i^{cheat(c,c_1)}(v') \,\middle|\, V_j^{abs} \,\middle|\, E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle z\rangle\right]\right]$$

and

$$vc.S'\left[P \,\middle|\, V_i^{cheat(c,c_1)}(v') \,\middle|\, V_j^{abs} \,\middle|\, E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle v\rangle\right]\right]$$

is to be proved. Figure 31-33, 71-83 give the inputs in extension of the PI calculus (Abadi and Blanchet, 2005) of verification of Condition 4 in ProVerif. The result shows that the observational equivalence between:

$$vc.S'\left[P \,\middle|\, V_i^{cheat(c,c_1)}(v') \,\middle|\, V_j^{abs} \,\middle|\, E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle z\rangle\right]\right]$$

and

$$vc.S'\left[P \,\middle|\, V_i^{cheat(c,c_1)}(v') \,\middle|\, V_j^{abs} \,\middle|\, E_k^{c_1,c_2,z}\left[\overline{c_{votes}}\langle v\rangle\right]\right]$$

is true in Fig. 84. Bad is not derivable shows that observational equivalence is true.

In condition 5 an additional restriction is introduced that justifies the abstraction of the third voter by the

```
let extractor =
  in(chVII,id);
  new nonceE;
  out(chVR3,(n1,nonceE));
  in(chVR3,(=n2,=nonceE,credI,credJ,credE));
  in(chVE,(coercercred,=credI,=credJ,=credE));
  ! in(chTE,(=n1,vencvote,enccred1));
  let encvote=TPKenc(va,PK(V)) in
  if true=equals(vencvote,(coercercred,encvote)) then out(pub,va) else
  if true=equals(vencvote,(choice[credJ,credI],encvote)) then out(pub,va).
```

Fig. 52: Coercion-resistance-condition2-extractor

```
let modified_tallying_authority =
  new nonce;
  out(chcRT,(n1,nonce));
  in(chcRT,(=n2,=nonce,enccred1));
  in(pub,res);
  let result=PKdec(res,SK(S)) in
  out(chTE,(n1,result,enccred1)).
```

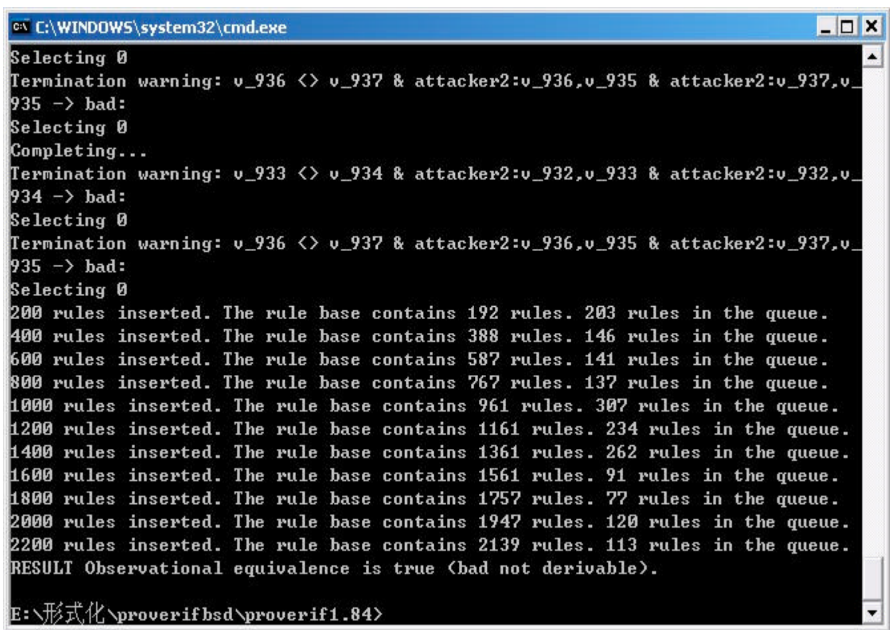
Fig. 53: Coercion-resistance-condition2-modified-tallying authority

```
let modified_registration_authority =
  in(chIIR,id);
  in(chcRT,(=n1,nonceT));
  in(chcRT,(=n1,nonceT2));
  in(chcRT,(=n1,nonceT3));
  in(chVR1,(=n1,nonceV));
  in(chVR2,(=n1,nonceV2));
  in(chVR3,(=n1,nonceE));
  new credI;
  new credJ;
  new credE;
  new m;
  out(chcRT,(n2,nonceT,TPKenc(choice[credI,credE],PK(V))));
  out(chcRT,(n2,nonceT2,TPKenc(choice[credJ,credI],PK(V))));
  out(chcRT,(n2,nonceT3,TPKenc(choice[credE,credJ],PK(V))));
  out(chVR1,(n2,nonceV,PKenc(zk((credI,sign(m,voter)),(TPKenc(credI,PK(V)),PK(V),m,VK(voter))),
      PK(voter))));
  out(chVR2,(n2,nonceV2,TPKenc(credI,PK(V)),TPKenc(credJ,PK(V)),TPKenc(credE,PK(V))));
  out(chVR3,(n2,nonceE,TPKenc(credI,PK(V)),TPKenc(credJ,PK(V)),TPKenc(credE,PK(V)))).
```

Fig. 54: Coercion-resistance-condition2-modified registration authority

```
process new C;new V;new S;
  new voter;
  new chVR;
  new chVR1;
  new chVR2;
  new chVR3;
  new chVII;
  new chIIR;
  new chRT;
  new chcRT;
  out(pub,PK(C));
  out(pub,PK(V));
  out(pub,PK(S));
  out(pub,PK(voter));
  (!voter|!corruptedvoter|!tallying_authority|!registration_authority
  |!issuer_authority|!votechooser |! modified_tallying_authority
  |modified_registration_authority | votercast | voterreg | coercedvoter | extractor)
```

Fig. 55: Coercion-resistance-condition2-process


```
C:\WINDOWS\system32\cmd.exe
Selecting 0
Termination warning: v_936 <> v_937 & attacker2:v_936,v_935 & attacker2:v_937,v_
935 -> bad:
Selecting 0
Completing...
Termination warning: v_933 <> v_934 & attacker2:v_932,v_933 & attacker2:v_932,v_
934 -> bad:
Selecting 0
Termination warning: v_936 <> v_937 & attacker2:v_936,v_935 & attacker2:v_937,v_
935 -> bad:
Selecting 0
200 rules inserted. The rule base contains 192 rules. 203 rules in the queue.
400 rules inserted. The rule base contains 388 rules. 146 rules in the queue.
600 rules inserted. The rule base contains 587 rules. 141 rules in the queue.
800 rules inserted. The rule base contains 767 rules. 137 rules in the queue.
1000 rules inserted. The rule base contains 961 rules. 307 rules in the queue.
1200 rules inserted. The rule base contains 1161 rules. 234 rules in the queue.
1400 rules inserted. The rule base contains 1361 rules. 262 rules in the queue.
1600 rules inserted. The rule base contains 1561 rules. 91 rules in the queue.
1800 rules inserted. The rule base contains 1757 rules. 77 rules in the queue.
2000 rules inserted. The rule base contains 1947 rules. 120 rules in the queue.
2200 rules inserted. The rule base contains 2139 rules. 113 rules in the queue.
RESULT Observational equivalence is true (bad not derivable).

E:\形式化\proverifbsd\proverif1.84>
```

Fig. 56: The result of coercion-resistance-condition2

```
free pub,com.(*public channel*)
private free c,chvote,chTE,chVE,internal1,internal.
free va,vb.
free n1,n2.
(*voter*)
let votechooser = out(chvote,va)|out(chvote,vb) .
```

Fig. 57: Coercion-resistance-condition3-vote chooser

```
let voter=
    in(chVII,id);
    new nonce;
    out(chVR,(n1,nonce));
    in(chVR,(=n2,=nonce,ct));
    let zkp=PKdec(ct,SK(voter)) in
    if zkver(zkp)=true then
    let (enccred1,enccred2,=PK(V),m,vk)=public1(zkp) in
    if check(sign(m,voter),vk)=m then
    let cred=multi(enccred1,enccred2) in
    in(chvote,vote);
    new r1;new r2;
    let encvote=multi(TpPKenc(projection1(vote),PK(V),r1),TpPKenc(projection2(vote),PK(V),r2)) in
    let ballot=multi(cred,encvote) in
    let res=PKenc(ballot,PK(S)) in
    out(pub,res).
```

Fig. 58: Coercion-resistance-condition3-voter

```
let corruptedvoter=
    in(chVII,id);
    new nonce;
    out(chVR,(n1,nonce));
    in(chVR,(=n2,=nonce,ct));
    out(pub,ct).
```

Fig. 59: Coercion-resistance-condition3-corrupted voter

```
let voterreg =
    new nonce1;
    out(chVR2,(n1,nonce1));
    in(chVR2,(=n2,=nonce1,credI,credJ,credE));
    out(internal1,(n2,credI,credJ,credE)).
```

Fig. 60: Coercion-resistance-condition3-voter registration

```
let coercedvoter =
    in(chVII,id);
    new nonce;
    out(chVR1,(n1,nonce));
    in(chVR1,(=n2,=nonce,ct));
    let zkp=PKdec(ct,SK(voter)) in
    if zkver(zkp)=true then
    let (enccred,=PK(V),m,vk)=public1(zkp) in
    if check(sign(m,voter),vk)=m then
    let cred=enccred in
    new fakecred;
    out(internal,(n2,cred,fakecred)).
```

Fig. 61: Coercion-resistance-condition3-coerced voter

```
let votercast =
    in (internal,(=n2,credI,fakecred));
    in (internal1,(=n2,=credI,credJ,credE));
    out (chVE,(fakecred,credI,credJ,credE));
    let encvote=TPKenc(va,PK(V)) in
    let res=PKenc((credI,encvote),PK(S)) in
    out (pub,res).
```

Fig. 62: Coercion-resistance-condition3-voter casting

```
let tallying_authority=
    new nonce;
    out (chRT,(n1,nonce));
    in (chRT,(=n2,=nonce,enccred1,enccred2));
    in (pub,res);
    let cenccred=multi(enccred1,enccred2) in
    let result=PKdec(res,SK(S)) in
    new r1;new r2;
    let cencvotea=multi(TpPKenc(projection1(va),PK(C),r1),TpPKenc(projection2(va),PK(C),r2)) in
    let cencvoteb=multi(TpPKenc(projection1(vb),PK(C),r1),TpPKenc(projection2(vb),PK(C),r2)) in
    let test1=multi(cenccred,cencvotea) in
    let test2=multi(cenccred,cencvoteb) in
    if true=checkciphertext(test1,result,C,V) then out(com,va) else
    if true=checkciphertext(test2,result,C,V) then out(com,vb).
```

Fig. 63: Coercion-resistance-condition3-tallying authority

```
let registration_authority=
    in(chIIR,id);
    in(chVR,(=n1,nonceV));
    in(chRT,(=n1,nonceT));
    new cred;
    let cred1=projection1(cred) in
    let cred2=projection2(cred) in
    new r1;new r2;
    out(chRT,(n2,nonceT,TpPKenc(cred1,PK(C),r1),TpPKenc(cred2,PK(C),r2)));
    new m;new r3;new r4;
    out(chVR,(n2,nonceV,PKenc(zk((cred1,cred2,sign(m,voter)),(TpPKenc(cred1,PK(V),r3),
        TpPKenc(cred2,PK(V),r4),PK(V),m,VK(voter))),PK(voter)))).
```

Fig. 64: Coercion-resistance-condition3-registration authority

```
let issuer_authority=
    new id;
    out(chVII,id);
    out(chIIR,id);
    out(pub,id).
```

Fig. 65: Coercion-resistance-condition3-issuer authority

```
let extractor =
    in(chVII,id);
    new nonceE;
    out(chVR3,(n1,nonceE));
    in(chVR3,(=n2,=nonceE,credI,credJ,credE));
    in(chVE,(coercercred,=credI,=credJ,=credE));
    ! in(chTE,(=n1,vencvote,enccred1));
    let encvote=TPKenc(va,PK(V)) in
    if true=equals(vencvote,(choice[credI,credE],encvote)) then
    out(pub,choice[va,vb]).
```

Fig. 66: Coercion-resistance-condition3-extractor

```
let modified_tallying_authority =
    new nonce;
    out(chcRT,(n1,nonce));
    in(chcRT,(=n2,=nonce,enccred1));
    in(pub,res);
    let result=PKdec(res,SK(S)) in
    out(chTE,(n1,result,enccred1)).
```

Fig. 67: Coercion-resistance-condition3-modified tallying authority

```
let modified_registration_authority =
    in(chIIR,id);
    in(chcRT,(=n1,nonceT));
    in(chcRT,(=n1,nonceT2));
    in(chcRT,(=n1,nonceT3));
    in(chVR1,(=n1,nonceV));
    in(chVR2,(=n1,nonceV2));
    in(chVR3,(=n1,nonceE));
    new credI;
    new credJ;
    new credE;
    new m;
    out(chcRT,(n2,nonceT,TPKenc(credI,PK(V))));
    out(chcRT,(n2,nonceT2,TPKenc(credJ,PK(V))));
    out(chcRT,(n2,nonceT3,TPKenc(credE,PK(V))));
    out(chVR1,(n2,nonceV,PKenc(zk((credI,sign(m,voter)),
        (TPKenc(credI,PK(V)),PK(V),m,VK(voter))),PK(voter))));
    out(chVR2,(n2,nonceV2,TPKenc(credI,PK(V)),TPKenc(credJ,PK(V)),TPKenc(credE,PK(V))));
    out(chVR3,(n2,nonceE,TPKenc(credI,PK(V)),TPKenc(credJ,PK(V)),TPKenc(credE,PK(V)))).
```

Fig. 68: Coercion-resistance-condition3-modified registration authority

Extractor: votes with invalid registration secrets are silently discarded by the tallying authority. If this was not the case a coercer could easily distinguish real from fake registration secrets. The observational equivalence between:

$$S\left[V_i^{inv\text{-}reg}\right]$$

and

$$\nu c_{votes}.\left(!\,c_{votes}(x)\right)\middle|S\left[V_i(v)\right]$$

```
process new C;new V;new S;
    new voter;
    new chVR;
    new chVR1;
    new chVR2;
    new chVR3;
    new chVII;
    new chIIR;
    new chRT;
    new chcRT;
    out(pub,PK(C));
    out(pub,PK(V));
    out(pub,PK(S));
    out(pub,PK(voter));
    (!voter|!corruptedvoter|!tallying_authority|!registration_authority|!issuer_authority|!votechooser
    |! modified_tally_authority |modified_registration_authority | votercast | voterreg | coercedvoter | extractor)
```

Fig. 69: Coercion-resistance-condition3-process

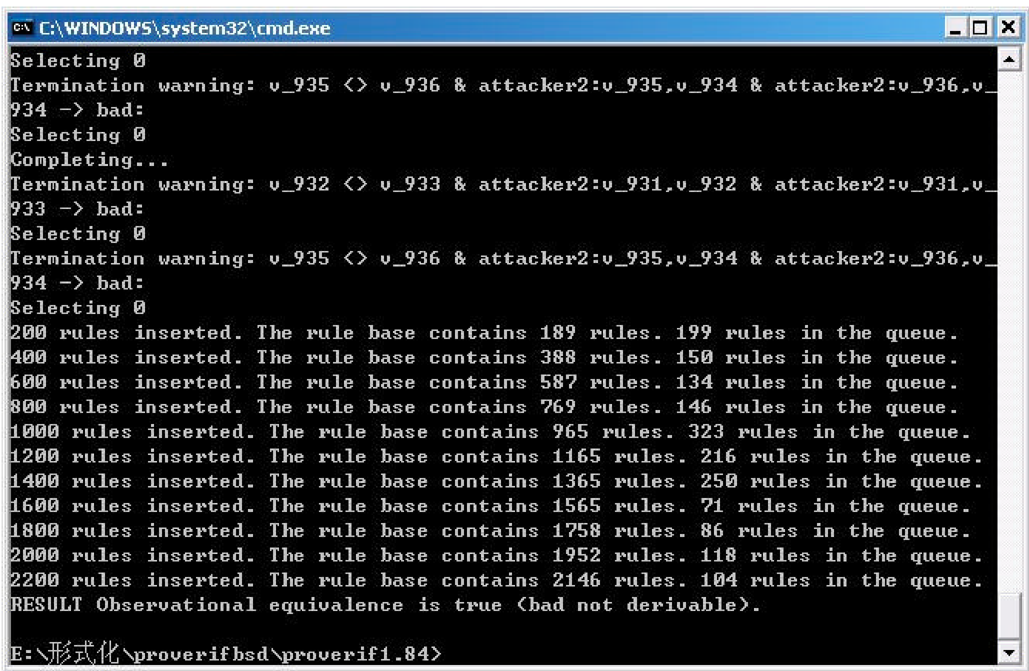


Fig. 70: The result of coercion-resistance-condition3

```
free pub,com.(*public channel *)
private free c,chvote,chTE,chVE,internal1,internal.
free va,vb.
free n1,n2.
(*voter *)
let votechooser = out(chvote,va) | out(chvote,vb).
```

Fig. 71: Coercion-resistance-condition4-vote choose

```
let voter=
    in(chVII,id);
    new nonce;
    out(chVR,(n1,nonce));
    in(chVR,(=n2,=nonce,ct));
    let zkp=PKdec(ct,SK(voter)) in
    if zkver(zkp)=true then
    let (enccred1,enccred2,=PK(V),m,vk)=public1(zkp) in
    if check(sign(m,voter),vk)=m then
    let cred=multi(enccred1,enccred2) in
    in(chvote,vote);
    new r1;new r2;
    let encvote=multi(TpPKenc(projection1(vote),PK(V),r1),TpPKenc(projection2(vote),PK(V),r2)) in
    let ballot=multi(cred,encvote) in
    let res=PKenc(ballot,PK(S)) in
    out(pub,res).
```

Fig. 72: Coercion-resistance-condition4-voter

```
let corruptedvoter=
    in(chVII,id);
    new nonce;
    out(chVR,(n1,nonce));
    in(chVR,(=n2,=nonce,ct));
    out(pub,ct).
```

Fig. 73: Coercion-resistance-condition4-corrupted voter

```
let voterreg =
    new nonce1;
    out(chVR2,(n1,nonce1));
    in(chVR2,(=n2,=nonce1,credI,credJ,credE));
    out(internal1,(n2,credI,credJ,credE)).
```

Fig. 74: Coercion-resistance-condition4-voter registration

```
let coercedvoter =
    in(chVII,id);
    new nonce;
    out(chVR1,(n1,nonce));
    in(chVR1,(=n2,=nonce,ct));
    let zkp=PKdec(ct,SK(voter)) in
    if zkver(zkp)=true then
    let (enccred,=PK(V),m,vk)=public1(zkp) in
    if check(sign(m,voter),vk)=m then
    let cred=enccred in
    new fakecred;
    out(internal,(n2,cred,fakecred)).
```

Fig. 75: Coercion-resistance-condition4-coerced voter

```
let votercast =
    in(internal,(=n2,credI,fakecred));
    in(internal1,(=n2,=credI,credJ,credE));
    out(chVE,(fakecred,credI,credJ,credE));
    let encvote=TPKenc(va,PK(V)) in
    let res=PKenc((credI,encvote),PK(S)) in
    out(pub,res)
    let encvote=TPKenc(vb,PK(V)) in
    let res=PKenc((fakecred,encvote),PK(S)) in
    out(pub,res).
```

Fig. 76: Coercion-resistance-condition4-voter casting

```
let tallying_authority=
    new nonce;
    out(chRT,(n1,nonce));
    in(chRT,(=n2,=nonce,enccred1,enccred2));
    in(pub,res);
    let cenccred=multi(enccred1,enccred2) in
    let result=PKdec(res,SK(S)) in
    new r1;new r2;
    let cencvotea=multi(TpPKenc(projection1(va),PK(C),r1),TpPKenc(projection2(va),PK(C),r2)) in
    let cencvoteb=multi(TpPKenc(projection1(vb),PK(C),r1),TpPKenc(projection2(vb),PK(C),r2)) in
    let test1=multi(cenccred,cencvotea) in
    let test2=multi(cenccred,cencvoteb) in
    if true=checkciphertext(test1,result,C,V) then out(com,va) else
    if true=checkciphertext(test2,result,C,V) then out(com,vb).
```

Fig. 77: Coercion-resistance-condition4-tallying authority

```
let registration_authority=
    in(chIIR,id);
    in(chVR,(=n1,nonceV));
    in(chRT,(=n1,nonceT));
    new cred;
    let cred1=projection1(cred) in
    let cred2=projection2(cred) in
    new r1;new r2;
    out(chRT,(n2,nonceT,TpPKenc(cred1,PK(C),r1),TpPKenc(cred2,PK(C),r2)));
    new m;new r3;new r4;
    out(chVR,(n2,nonceV,PKenc(zk((cred1,cred2,sign(m,voter)),(TpPKenc(cred1,PK(V),r3),
        TpPKenc(cred2,PK(V),r4),PK(V),m,VK(voter))),PK(voter)))).
```

Fig. 78: Coercion-resistance-condition4-registration authority

```
let issuer_authority=
    new id;
    out(chVII,id);
    out(chIIR,id);
    out(pub,id).
```

Fig. 79: Coercion-resistance-condition4-issuer authority

```
let extractor =
    in(chVII,id);
    new nonceE;
    out(chVR3,(n1,nonceE));
    in(chVR3,(=n2,=nonceE,credI,credJ,credE));
    in(chVE,(coercercred,=credI,=credJ,=credE));
    ! in(chTE,(=n1,vencvote,enccred1));
    let encvote=TPKenc(va,PK(V)) in
    if true=equals(vencvote,(choice[credI,credE],encvote)) then
    out(pub,choice[va,vb]).
```

Fig. 80: Coercion-resistance-condition4-extractor

```
let modified_tallying_authority =
    new nonce;
    out(chcRT,(n1,nonce));
    in(chcRT,(=n2,=nonce,enccred1));
    in(pub,res);
    let result=PKdec(res,SK(S)) in
    out(chTE,(n1,result,enccred1)).
```

Fig. 81: Coercion-resistance-condition4-modified tallying authority

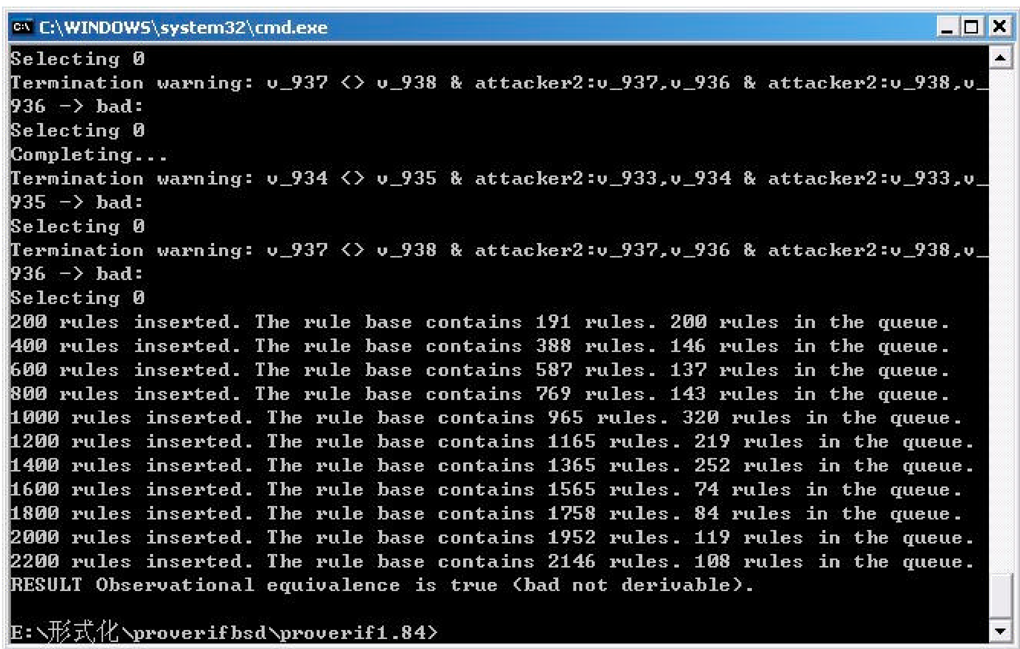
```
let modified_registration_authority =
    in(chIIR,id);
    in(chcRT,(=n1,nonceT));
    in(chcRT,(=n1,nonceT2));
    in(chcRT,(=n1,nonceT3));
    in(chVR1,(=n1,nonceV));
    in(chVR2,(=n1,nonceV2));
    in(chVR3,(=n1,nonceE));
    new credI;
    new credJ;
    new credE;
    new m;
    out(chcRT,(n2,nonceT,TPKenc(credI,PK(V))));
    out(chcRT,(n2,nonceT2,TPKenc(credJ,PK(V))));
    out(chcRT,(n2,nonceT3,TPKenc(credE,PK(V))));
    out(chVR1,(n2,nonceV,PKenc(zk((credI,sign(m,voter)),
        (TPKenc(credI,PK(V)),PK(V),m,VK(voter))),PK(voter))));
    out(chVR2,(n2,nonceV2,TPKenc(credI,PK(V)),TPKenc(credJ,PK(V)),TPKenc(credE,PK(v))));
    out(chVR3,(n2,nonceE,TPKenc(credI,PK(V)),TPKenc(credJ,PK(V)),TPKenc(credE,PK(v)))).
```

Fig. 82: Coercion-resistance-condition4-modified registration authority

is proved in ProVerif. Figure 31-33, 85-91 give the inputs in extension of the PI calculus (Abadi and Blanchet, 2005) of verification of Condition 2 in ProVerif. The result shows that the observational

```
process new C;new V;new S;
    new voter;
    new chVR;
    new chVR1;
    new chVR2;
    new chVR3;
    new chVII;
    new chIIR;
    new chRT;
    new chcRT;
    out(pub,PK(C));
    out(pub,PK(V));
    out(pub,PK(S));
    out(pub,PK(voter));
    (!voter|!corruptedvoter|!tallying_authority|!registration_authority|!issuer_authority|!votechooser
    |! modified_tally_authority |modified_registration_authority | votercast | voterreg | coercedvoter | extractor)
```

Fig. 83: Coercion-resistance-condition4-process



```
C:\WINDOWS\system32\cmd.exe
Selecting 0
Termination warning: v_937 <> v_938 & attacker2:v_937,v_936 & attacker2:v_938,v_
936 -> bad:
Selecting 0
Completing...
Termination warning: v_934 <> v_935 & attacker2:v_933,v_934 & attacker2:v_933,v_
935 -> bad:
Selecting 0
Termination warning: v_937 <> v_938 & attacker2:v_937,v_936 & attacker2:v_938,v_
936 -> bad:
Selecting 0
200 rules inserted. The rule base contains 191 rules. 200 rules in the queue.
400 rules inserted. The rule base contains 388 rules. 146 rules in the queue.
600 rules inserted. The rule base contains 587 rules. 137 rules in the queue.
800 rules inserted. The rule base contains 769 rules. 143 rules in the queue.
1000 rules inserted. The rule base contains 965 rules. 320 rules in the queue.
1200 rules inserted. The rule base contains 1165 rules. 219 rules in the queue.
1400 rules inserted. The rule base contains 1365 rules. 252 rules in the queue.
1600 rules inserted. The rule base contains 1565 rules. 74 rules in the queue.
1800 rules inserted. The rule base contains 1758 rules. 84 rules in the queue.
2000 rules inserted. The rule base contains 1952 rules. 119 rules in the queue.
2200 rules inserted. The rule base contains 2146 rules. 108 rules in the queue.
RESULT Observational equivalence is true (bad not derivable).

E:\形式化\proverifbsd\proverif1.84>
```

Fig. 84: The result of coercion-resistance-condition4

```
free pub,com.(*public channel *)
private free c,chvote,chTE,chVE,internal1,internal.
free va,vb.
free n1,n2.
(*voter *)
let votechooser = out(chvote,va) | out(chvote,vb).
```

Fig. 85: Coercion-resistance-condition5-vote chooser

```
let voter=
    in(chVII,id);
    new nonce;
    out(chVR,(n1,nonce));
    in(chVR,(=n2,=nonce,ct));
    let zkp=PKdec(ct,SK(voter)) in
    if  zkver(zkp)=true then
    let (enccred1,enccred2,=PK(V),m,vk)=public1(zkp) in
    if check(sign(m,voter),vk)=m then
    let cred=multi(enccred1,enccred2) in
    in(chvote,vote);
    new r1;new r2;
    let encvote=multi(TpPKenc(projection1(vote),PK(V),r1),TpPKenc(projection2(vote),PK(V),r2)) in
    let ballot=multi(cred,encvote) in
    let res=PKenc(ballot,PK(S)) in
    out(pub,res).
```

Fig. 86: Coercion-resistance-condition5-voter

```
let corruptedvoter=
    in(chVII,id);
    new nonce;
    out(chVR,(n1,nonce));
    in(chVR,(=n2,=nonce,ct));
    out(pub,ct).
```

Fig. 87: Coercion-resistance-condition5-corrupted voter

```
let voterchoice =
    new nonce;
    out(chVR,(n1,nonce));
    in(chVR,(=n2,=nonce,ct));
    let zkp=PKdec(ct,SK(voter)) in
    if  zkver(zkp)=true then
    let (enccred1,enccred2,=PK(V),m,vk)=public1(zkp) in
    if check(sign(m,voter),vk)=m then
    let cred=multi(enccred1,enccred2) in
    in(chvote,vote);
    new r1;new r2;
    new fakecred;
    let encvote=multi(TpPKenc(projection1(vote),PK(V),r1),TpPKenc(projection2(vote),PK(V),r2)) in
    let ballot=multi(choice[cred,fakecred],encvote) in
    let res=PKenc(ballot,PK(S)) in
    out(pub,res).
```

Fig. 88: Coercion-resistance-condition5-voter choice

equivalence is true in Fig. 92. Bad is not derivable shows that observational equivalence is true.

According to the above analysis we can found that the Acquisti protocol has the coercion-resistance

```
let tallying_authority=
    new nonce;
    out(chRT,(n1,nonce));
    in(chRT,(=n2,=nonce,enccred1,enccred2));
    in(pub,res);
    let cenccred=multi(enccred1,enccred2) in
    let result=PKdec(res,SK(S)) in
    new r1;new r2;
    let cencvotea=multi(TpPKenc(projection1(va),PK(C),r1),TpPKenc(projection2(va),PK(C),r2)) in
    let cencvoteb=multi(TpPKenc(projection1(vb),PK(C),r1),TpPKenc(projection2(vb),PK(C),r2)) in
    let test1=multi(cenccred,cencvotea) in
    let test2=multi(cenccred,cencvoteb) in
    if true=checkciphertext(test1,result,C,V) then out(com,va) else
    if true=checkciphertext(test2,result,C,V) then out(com,vb).
```

Fig. 89: Coercion-resistance-condition5-tallying authority

```
let registration_authority=
    in(chIIR,id);
    in(chVR,(=n1,nonceV));
    in(chRT,(=n1,nonceT));
    new cred;
    let cred1=projection1(cred) in
    let cred2=projection2(cred) in
    new r1;new r2;
    out(chRT,(n2,nonceT,TpPKenc(cred1,PK(C),r1),TpPKenc(cred2,PK(C),r2)));
    new m;new r3;new r4;
    out(chVR,(n2,nonceV,PKenc(zk((cred1,cred2,sign(m,voter)),(TpPKenc(cred1,PK(V),r3),
        TpPKenc(cred2,PK(V),r4),PK(V),m,VK(voter))),PK(voter)))).
```

Fig. 90: Coercion-resistance-condition5-registration authority

```
process new C;new V;new S;
    new voter;
    new chVR;
    new chRT;
    out(pub,PK(C));
    out(pub,PK(V));
    out(pub,PK(S));
    out(pub,PK(voter));
    (!voter|!corruptedvoter|!tallying_authority|!registration_authority|!votechooser |voterchoice)
```

Fig. 91: Coercion-resistance-condition5-process

with the assumption that the channel chVR1, chVR2 and chVR3 between modified registration authority and coerced voter is private channel. If these channels are public then the coercer could easily distinguish real from fake registration secrets, thus the condition2 of coercion-resistance is not satisfied. The result is showed in Fig. 93.

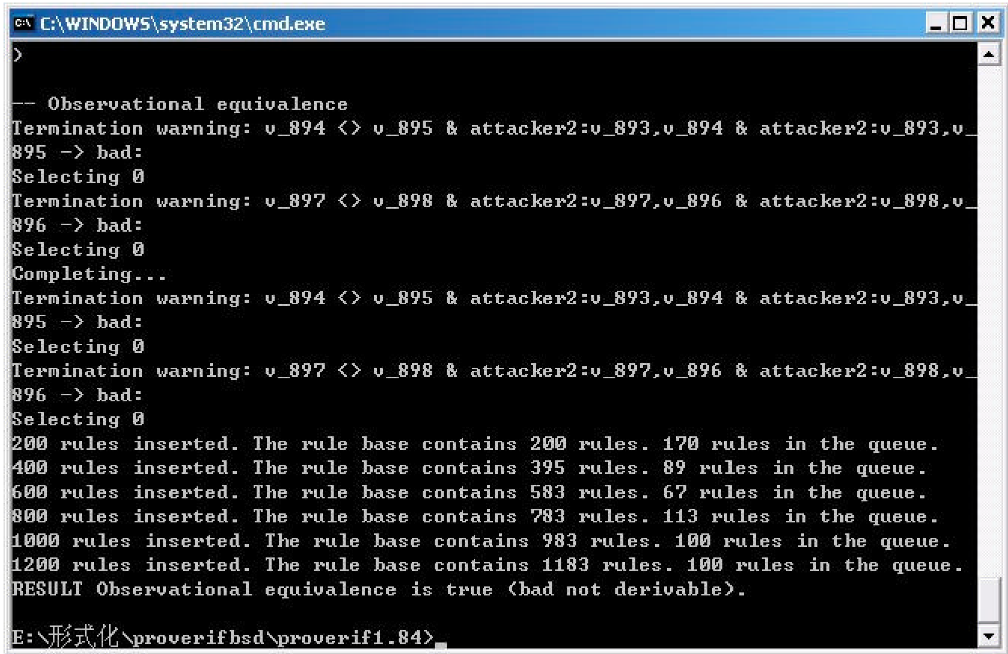


Fig. 92: The result of coercion-resistance-condition5

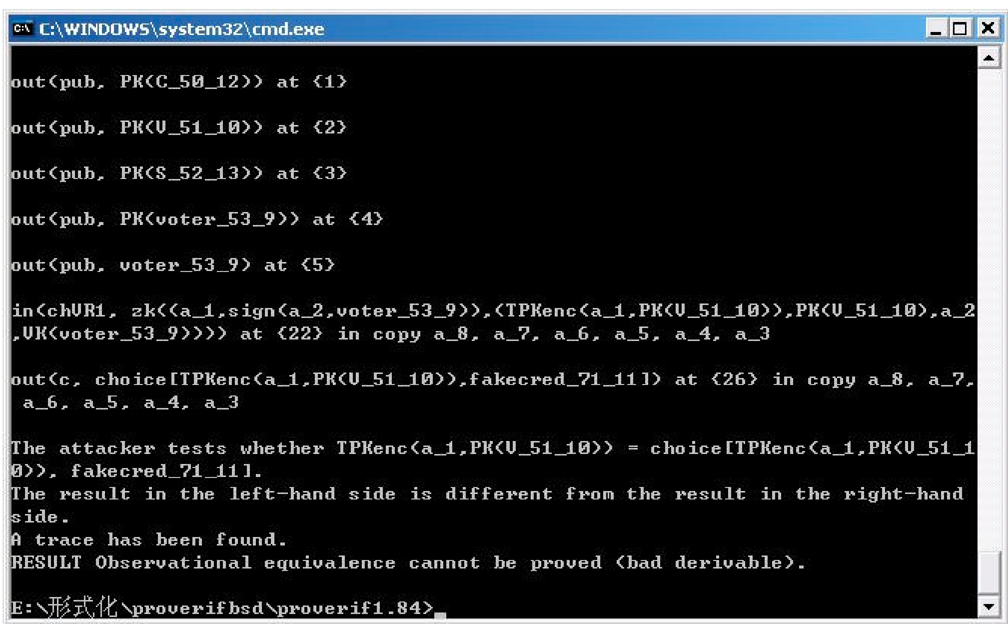


Fig. 93: The result of coercion-resistance-condition2 with the condition that the channels are public

CONCLUSION AND FUTURE WORK

Internet voting protocol play an important role in remote voting system. Acquisti protocol is one of the most important remote internets voting protocol that claims to satisfy formal definitions of key properties, such as soundness, individual verifiability, as well as receipt-freeness and coercion resistance without strong physical constrains. But the analysis of its claimed security properties is finished by hand which depends on experts' knowledge and skill and is prone to make mistakes. Recently owning to the contribution of Backes *et al*. (2008a) Acquisti protocol can be proved with mechanized proof tool ProVerif. In this study the review of the formal method of electronic voting protocols are introduced we found that several formal models have been proposed, but only the Backes *et al*. (2008b) model supports the mechanized proof tool; the formal model and proof of security properties mainly focus on receipt-freeness and coercion-resistance which are important properties. Until now people have not proposed a security analysis model based on computational model, then applied PI calculus and the mechanized proof tool ProVerif are examined. After that Acquisti protocol is modeled in applied PI calculus. Security properties, including soundness and coercion resistance, are verified with ProVerif. The result we obtain is that Acquisti protocol has the soundness. At the same time it has also coercion-resistance in the conditions that the channel between registration authority and voter is private. To our best knowledge, the first mechanized proof of Acquisti protocol for an unbounded number of honest and corrupted voters is finished.

As future work, we plan to prove other internet voting protocols. It would also be interesting to formalize the security properties in wireless communication protocol in the formal model with mechanized proof tool ProVerif. At the same time we will formalize the security properties of remote internet voting protocols in the computational model with mechanized tool CryptoVerif.

ACKNOWLEDGMENT

This study was supported in part by Natural Science Foundation of South-Center University for Nationalities under the grants No: YZZ06026, titled Research on the internet voting protocols with receipt-freeness, conducted in Wuhan, China from 06/11/2006 to 20/11/2009. The preliminary part work in this study is published in Meng *et al*. (2010c).

REFERENCES

Abadi, M. and A.D. Gordon, 1999. A calculus for cryptographic protocols: The spi calculus. Inform. Comput., 148: 1-70.

Abadi, M. and C. Fournet, 2001. Mobile values, new names and secure communication. Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK., March 2001, ACM New York, USA., pp: 104-115.

Abadi, M., B. Blanchet and C. Fournet, 2004. Just Fast Keying in the PI Calculus. In: Programming Languages and Systems, Sc hmidt, D. (Ed.). LNCS., 2986, Springer, Berlin, Heidelberg, ISBN-13: 978-3-540-21313-0, pp: 340-354.

Abadi, M. and B. Blanchet, 2005. Analyzing security protocols with secrecy types and logic programs. J. ACM, 52: 102-146.

Acquisti, A., 2004. Receipt-free homomorphic elections and write-in voter verified ballot. CMU-ISRI-04-116, 2004, Carnegie Mellon Institute for Software Research International. http://www.heinz.cmu.edu/~acquisti/papers/acquisti-electronic_voting.pdf.

Aditya, R., B. Lee, C. Boyd and E. Dawson, 2004. An efficient mixnet-based voting scheme providing receipt-freeness. Lecture Notes Comput. Sci., 3184: 152-161.

Backes, M., C. Hritcu and M. Maffei, 2008a. Automated verification of remote electronic voting protocols in the applied Pi-calculus. Proceedings of the 21st IEEE Computer Security Foundations Symposium, June 23-25, IEEE Computer Society, Washington, DC, pp: 195-209.

Backes, M., M. Maffei and D. Unruh, 2008b. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. Proceedings of the 29th IEEE Symposium on Security and Privacy, May 2008, Preprint on IACR ePrint, pp: 202-215.

Baskar, A., R. Ramanujam and S.P. Suresh, 2007. Knowledge-based modelling of voting protocols. Proceedings of the 11th Conference on theoretical Aspects of Rationality and Knowledge, June 25-27, Brussels, Belgium, pp: 62-71.

Baudron, O., P.A. Fouque, D. Pointcheval, G. Poupard and S. Jacques, 2001. Practical multi-candidate election system. Proceedings of the Annual ACM Symposium on Principles of Distributed Computing, 2001, ACM, New York, USA., pp: 274-283.

Benaloh, J. and D. Tuinstra, 1994. Receipt-free secret-ballot elections. Proceeding of the 26th Annual ACM Symposium on Theory of Computing, May 23-25, ACM, New York, USA., pp: 544-553.

Bhargavan, K., R. Corin, C. Fournet and E. Zalinescu, 2008. Cryptographically verified implementations for TLS. Proceedings of the 15th ACM Conference on Computer and Communications Security, Oct. 27-31, Alexandria, Virginia, USA.., pp: 459-468.

Blanchet, B., 2001. An efficient cryptographic protocol verifier based on prolog rules. Proceedings of the 14th IEEE Workshop on Computer Security Foundations, Jun. 11-13, IEEE Computer Society, Washington, DC, pp: 82-96.

Blanchet, B., 2008. A computationally sound mechanized prover for security protocols. IEEE Trans. Dependable Secure Comput., 5: 193-207.

Blum, M. and S. Micali, 1984. How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. Comput., 13: 850-864.

Burrows, M., M. Abadi and R. Needham, 1989. A logic of authentication. SIGOPS Operat. Syst. Rev., 23: 1-13.

Burrows, M., M. Abadi and R. Needham, 1990. A logic of authentication. ACM Trans. Comput. Syst., 8: 18-36.

Chaum, D.L., 1981. Untraceable electronic mail, return addresses and digital pseudonyms. Commun. ACM, 24: 84-88.

Chaum, D., 2004. Secret-ballot receipts: True voter-verifiable elections. IEEE Security Privacy, 2: 38-47.

Chaum, D., P. Y.A. Ryan and S. Schneider, 2005. A practical voter-verifiable election scheme. Proceedings of the ESORICS, Sept. 12-14, Milan, Italy, pp: 118-139.

Cichon, J., M. Kutylowski and B. Glorz, 2008. Short Ballot Assumption and Threeballot Voting Protocol. In: SOFSEM 2008: Theory and Practice of Computer Science, Geffert, V. *et al*. (Eds.). Springer-Verlag, Berlin Heidelberg, pp: 585-598.

Clarke, E.M., S. Jha and W. Marrero, 2000. Verifying security protocols with Brutus. ACM Trans. Softw. Eng. Methodol., 9: 443-487.

Clarkson, M.R., S. Chong and A.C. Myers, 2008. Civitas: Toward a secure voting system. Proceeding of the 2008 IEEE Symposium on Security and Privacy, May 18-21, Oakland, California, USA., pp: 354-368.

Cramer, R., R. Gennaro and B. Schoenmakers, 1997. A Secure and Optimally Efficient Multi-Authority Election Scheme. In: Trustworthy Global Computing, Fumy, W. (Ed.). Springer-Verlag, Berlin Heidelberg, pp: 103-118.

DeMillo, R.A., N.A. Lynch and M.J. Merritt, 1982. Cryptographic protocols. Proceedings of the 14th Annual ACM Symposium on theory of Computing, May 05-07, San Francisco, California, United States, pp: 383-400.

Delaune, S., S. Kremer and M. Ryan, 2005. Receipt-freeness: Formal definition and fault attacks. http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/DKR-fee05.pdf.

Delaune, S., S. Kremer and M.D. Ryan, 2006a. Coercion-resistance and receipt-freeness in electronic voting protocol. Proceedings of 19th IEEE Computer Security Foundations Workshop, July 5-7, Venice, Italy, pp: 28-42.

Delaune, S., S. Kremer and M. Ryan, 2006b. Verifying properties of electronic voting protocols. http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/DKR-wote06.pdf.

Dolev, D. and A.C. Yao, 1983. On the security of public key protocols. IEEE Trans. Inform. Theor., 29: 198-208.

Fujioka, A., T. Okamoto and K. Ohta, 1992. A practical secret voting scheme for large-scale elections. Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology, December 13-16, Springer-Verlag, London, UK., pp: 244-251.

Gerling, S., D. Jednoralski and X.Y. GU, 2008. Towards the verification of the civitas remote electronic voting protocol using proverif. http://www.infsec.cs.uni-sb.de/teaching/WS07/Seminar/reports/civitas-proverif.pdf.

Groth, J., 2004. Evaluating Security of Voting Schemes in the Universal Composability Framework. In: Applied Cryptography and Network Security, Jakobsson, M., M. Yung and J. Zhou (Eds.). Springer-Verlag, Berlin Heidelberg, pp: 46-60.

Hirt, M. and K. Sako, 2000. Efficient receipt-free voting based on homomorphic encryption. Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, May 14-18, Bruges, Belgium, pp: 539-556.

Hoare, C.A., 1985. Communicating Sequential Processes. Prentice-Hall, Inc., USA.

Hubbers, E., B. Jacobs and W. Pieters, 2005. RIES-internet voting in action. Proceedings of the 29th Annual International Computer Software and Applications Conference, July 26-28, IEEE Computer Society, Washington, DC., pp: 417-424.

Jakobsson, M., K. Sako and R. Impagliazzo, 1996. Designated verifier proofs and their applications. Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, May 12-16, Saragossa, Spain, pp: 143-154.

Jonker, H.L. and E.P. de Vink, 2006. Formalising receipt-freeness. Proceedings of the 9th International Conference on Information Security, Aug. 30-Sept. 2, Samos Island, Greece, pp: 476-488.

Jonker, H.L. and W. Pieters, 2006. Receipt-freeness as a special case of anonymity in epistemic logic. Proceedings of the IAVoSS Workshop on Trustworthy Elections, June 29-30, 2006, Cambridge, UK.

Joseph, C. and F. Cremers, 2006. Scyther-semantics and verification of security protocols. http://alexandria.tue.nl/extra2/200612074.pdf.

Juels, A. and M. Jakobsson, 2002. Coercion-resistant electronic elections, 2002. http://www.vote-auction.net/VOTEAUCTION/165.pdf.

Juels, A., D. Catalano and M. Jakobsson, 2005. Coercion-resistant electronic elections. Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, Nov. 07-07, Alexandria, VA, USA., pp: 61-70.

Kessler, V. and H. Neumann, 1998. A sound logic for analysing electronic commerce protocols. Proceedings of the 5th European Symposium on Research in Computer Security, Sept. 16-18, London, UK., pp: 345-360.

Kindred, D., 1999. Theory generation for security protocols. Doctoral Thesis, Carnegie Mellon University.

Lee, B., C. Boyd, E. Dawson, K. Kim, J. Yang and S. Yoo, 2003. Providing receipt-freeness in mixnet-based voting protocols. http://caislab.icu.ac.kr/Paper/paper_files/2003/ICISC03/mnvoting-final-icisc20.pdf.

Lowe, G., 1998. Casper: A complier for the analysis of security protocols. J. Comput. Sec., 6: 53-84.

Maggi, P. and R. Sisto, 2002. Using SPIN to verify security properties of cryptographic protocols. Proceedings of the 9th international SPIN Workshop on Model Checking of Software, April 11-13, Springer-Verlag, London, pp: 187-204.

Magkos, E., M. Burmester and V. Chrissikopoulos, 2001. Receipt-freeness in large-scale elections without untappable channels. Proceedings of the IFIP Conference on Towards the E-Society: E-Commerce, E-Business, E-Government, Oct. 03-05, uwer B.V., Deventer, The Netherlands, pp: 683-694.

Mauw, S., J. Verschuren and E.P. De Vink, 2007. Data anonymity in the FOO voting scheme. Elect. Notes Theor. Comput. Sci., 168: 5-28.

McMillan, K.L., 1992. Symbolic model checking: An approach to the state explosion problem. Ph.D. Thesis, Carnegie Mellon University.

Meadows, C.A., 1996. The NRL protocol analyzer: An overview. J. Logic Programming, 26: 113-131.

Mei, J., H. Miao and P. Liu, 2009. Applying SMV for security protocol verification. Infom. Technol. J., 8: 1065-1070.

Meng, B., 2007a. Analysis of internet voting protocols with jonker-vink receipt freeness formal model. Proceedings of the International Conference on Convergence Information Technology, Nov. 21-23, ICCIT., IEEE Computer Society, Washington, DC., pp: 663-669.

Meng, B., 2007b. An internet voting protocol with receipt-free and coercion-resistant. Proceedings of 7th IEEE International Conference on Computer and Information Technology, Oct. 16-19, IEEE Computer Society, Washington DC, USA., pp: 721-726.

Meng, B., 2008. Formal analysis of key properties in the internet voting protocol using applied PI calculus. Inform. Technol. J., 7: 1133-1140.

Meng, B., 2009a. A secure internet voting protocol based on non-interactive deniable authentication protocol and proof protocol that two ciphertexts are encryption of the same plaintext J. Networks, 4: 370-377.

Meng, B., 2009b. A formal logic framework for receipt-freeness in internet voting protocol. J. Comput., 4: 184-192.

Meng, B., 2009c. A critical review of receipt-freeness and coercion-resistance. Inform. Technol. J., 8: 934-964.

Meng, B., 2009d. A secure non-interactive deniable authentication protocol with strong deniability based on discrete logarithm problem and its application on Internet voting protocol. Inform. Technol. J., 8: 302-309.

Meng, B. and J.Q. Wang, 2010. An efficient receiver deniable encryption scheme and its applications. J. Networks, 5: 683-690.

Meng, B., Z.M. Li and J. Qin, 2010. A receipt-free coercion-resistant remote internet voting protocol without physical assumptions through deniable encryption and trapdoor commitment scheme. J. Software, 5: 942-949.

Meng, B., W. Huang and J. Qin, 2010b. Automatic verification of security properties of remote internet voting protocol in symbolic model. Inform. Technol. J., 9: 1521-1556.

Meng, B., W. Huang and D.J. Wang, 2010c. Automatic verification of remote internet voting protocol in symbolic model. Recent Advances in Electronic Commerce and Information Technology of ISECS.

Merritt, M.J., 1983. Cryptographic protocols. Ph.D. Thesis, Georgia Institute of Technology.

Mitchell, J.C., M. Mitchell and U. Stern, 1997. Automated analysis of cryptographic protocols using Mur. Proceedings of the 1997 IEEE Symposium on Security and Privacy, May 04-07, Digital Library, pp: 141-141.

Paillier, P., 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Advances in Cryptology-EUROCRYPT '99, Stern, J. (Ed.). Springer-Verlag, Berlin Heidelberg, pp: 223-238.

Paulson, L.C., 1998. The inductive approach to verifying cryptographic protocols. Comput. Security, 6: 85-128.

Rivest, R.L., 2006. The threeBallot voting system. http://theory.csail.mit.edu/~rivest/Rivest-TheThreeBallotVotingSystem.pdf.

Sako, K. and J. Kilian, 1995. Receipt-Free Mix-Type Voting Scheme, A Practical Solution to the Implementation of a Voting Booth. In: Advances in Cryptology-EUROCRYPT '95, Guillou, L.C. and J.J. Quisquater (Eds.). Springer-Verlag, Berlin Heidelberg, pp: 393-403.

Song, D.X., 1999. Athena: A new efficient automatic checker for security protocol analysis. Proceedings of the 12th IEEE Workshop on Computer Security Foundations, June 28-30, IEEE Computer Society, Washington, DC., pp: 192-202.

Talbi, M., B. Morin, V. Viet Triem Tong, A. Bouhoula and M. Mejri, 2008. Specification of electronic voting protocol properties using ADM logic: FOO case study. Proceedings of the 10th international Conference on information and Communications Security, Oct. 20-22, Birmingham, UK., pp: 403-418.

Thayer, F., J.C. Herzog and J.D. Guttman, 1998. Strand space: Why is a security protocol correct? Proceedings of the 1998 IEEE Symposium on Security and Privacy, 1998, ACM, USA., pp: 160-171.

Van Eijck, J. and S. Orzan, 2007. Epistemic verification of anonymity. Elect. Notes Theor. Comput. Sci., 168: 159-174.

Yao, A.C., 1982. Theory and application of trapdoor functions. Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, Nov. 3-5, IEEE Computer Society, Washington, DC., pp: 80-91.