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 invariance. 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 (Lowc, 1998), Isabelle (Paulson, 1998), Athena (Song, 1999), Revere (Kindred, 1999), SPIN (Maggi and Sisto, 2002), Brutus (Clarke et al., 200), ProVerif (Blanchet, 2001), Seyther (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 Diffie-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.en.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).

Owing 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 (Lowc, 1998), Isabelle (Paulson, 1998), Athena (Song, 1999), Revere (Kindred, 1999), SPIN (Maggi and Sisto, 2002), Brutus (Clarke et al., 2000), ProVerif (Blanchet, 2001), Seyther (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 coeer to sharing secrets, but the coeer 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 coeer requests a given vote then \( V_b \) can change his vote and counterbalance the outcome. However, avoid the case where \( v' = V_a \cdot (c/v) \cdot v_b \) letting \( V_b \) vote \( u \). Therefore, requirement that when we apply a context \( C \), intuitively the coeer, requesting \( V_a \cdot (c/v) \cdot v_b \) to vote \( c \), \( v' \) in the same context votes \( u \). There may be circumstances where \( v' \) may need not to cast a vote that is not. In the case of coercion-resistance, the coeer 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 coeer 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 (Fujjoka 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 Kilian, 1995; Hilt and Sako, 2000; Aditya et al., 2004; Hubbers et al., 2005). Meng (2007b) analyzes receipt-freeness of the protocols (Fujjoka et al., 1992; Cramer et al., 1997; Juels and Jakobsen, 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: Rept ?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 (Fujjoka 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 Fujjoka 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 Fujjoka 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 (Fujikawa et al., 1992). Talbi et al. (2008) use ADM logic to specify security properties (fairness, eligibility, individual verifiability and universal verifiability) and analyze Fujikawa 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 find 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-resistance that are important properties.

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

<table>
<thead>
<tr>
<th>Properties</th>
<th>Formal method\footnote{applied PL calculus}</th>
<th>Delaune et al. \footnote{(2006a)}</th>
<th>Jonker and de Vink \footnote{(2006)}</th>
<th>Meng \footnote{(2006b)}</th>
<th>Jonker and Pieters \footnote{(2006)}</th>
<th>Bakes et al. \footnote{(2007)}</th>
<th>Bakes et al. \footnote{(2008a)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soundness</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Receipt-freeness</td>
<td>Applied PL calculus</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>Process algebra</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>Kessler and Neumann logic</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>Epistemic logic</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>Knowledge-based logic</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Coercion-resistance</td>
<td>Applied PL calculus</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

The mark \* represents the formal method is used. The mark \(\circ\) represents the formal models is supported by mechanized proof tool.

Table 2: The security properties formally modeled

<table>
<thead>
<tr>
<th>Properties</th>
<th>Fairness</th>
<th>Soundness</th>
<th>Eligibility</th>
<th>Privacy</th>
<th>Receipt-freeness</th>
<th>Coercion-resistance</th>
<th>Secrecy</th>
<th>verifiability</th>
<th>verifiability</th>
<th>Anonymity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bakes et al. \footnote{(2007)}</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Meng \footnote{(2006b)}</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Jonker and de Vink \footnote{(2006)}</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Delaune et al. \footnote{(2005)}</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Bakes et al. \footnote{(2008)}</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Talbi et al. \footnote{(2006b)}</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Manco et al. \footnote{(2007)}</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

The mark \* represents the property is formally defined; The mark \(\circ\) represents the formal definitions is supported by mechanized proof tool.

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

| Analyzed protocol | Bakes et al. \footnote{(2008b)} | Gerling et al. \footnote{(2008)} | Meng et al. \footnote{(2006b)} | Bakes et al. \footnote{(2008c)} | Meng et al. \footnote{(2007a)} | Meng et al. \footnote{(2007b)} | Meng et al. \footnote{(2007c)} | Benaloh and Tuinstra \footnote{(1994)} | Delaune et al. \footnote{(2005)} | Lee et al. \footnote{(2003)} |\footnote{analyzed protocol} |
|-------------------|----------------------------------|----------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|
|                   | Juels et al. \footnote{(2005)} | Clarke et al. \footnote{(2008)} | Meng et al. \footnote{(2004a)} | Juels and Jakobsson \footnote{(2002)} | Juels and Jakobsson \footnote{(2003)} | Juels and Jakobsson \footnote{(2004)} | Juels and Jakobsson \footnote{(2005)} | Benaloh and Tuinstra \footnote{(1994)} | Delaune et al. \footnote{(2005)} | Lee et al. \footnote{(2003)} |\footnote{analyzed protocol} |
|                   | \*                               | \*                               | \*                            | \*                            | \*                            | \*                            | \*                            | \*                                | \*                                | \*                                |

The mark \* represents the protocol has the property. The mark \(\square\) represents the protocol has not the property; The mark \(\Delta\) 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 Bakes et al. (2008a) to prove Acquisiti protocol and its security properties including soundness and coercion-resistance. Therefore, first, Acquisiti protocol is modeled in applied PI calculus and then its proof is performed by mechanized proof tool ProVerif. The result is that Acquisiti 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 Acquisiti protocol for an unbounded number of honest and corrupted voters.

Acquisiti 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. Acquisiti 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 Acquisiti 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 $\Sigma$. $\Sigma$ 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 $M_1, \ldots, M_n$ to $M$. 

297
Fig. 3: Terms

<table>
<thead>
<tr>
<th>M, N, T, V :=</th>
<th>terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>variable</td>
</tr>
<tr>
<td>a, b, c, ..., m, n</td>
<td>name</td>
</tr>
<tr>
<td>f(M, L, M)</td>
<td>function application</td>
</tr>
</tbody>
</table>

Fig. 4: Plain process

<table>
<thead>
<tr>
<th>P, Q, R :=</th>
<th>plain processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>null process</td>
</tr>
<tr>
<td>Q</td>
<td>P</td>
</tr>
<tr>
<td>!P</td>
<td></td>
</tr>
<tr>
<td>vn.P</td>
<td></td>
</tr>
<tr>
<td>if M = N then P else Q</td>
<td>conditional</td>
</tr>
<tr>
<td>in(u, x).P</td>
<td>message input</td>
</tr>
<tr>
<td>out(u, N).P</td>
<td>message output</td>
</tr>
</tbody>
</table>

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 vn is a sequence of name restrictions vn1, ..., vnn. 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, M).P for the input of terms M1, ..., Mn. 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, N).P is the abbreviation for the output of terms N1, ..., Nn. 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 \( \varphi \) and \( \psi \) range over frames. The domain dom(\( \varphi \)) of a frame \( \varphi \) is the set of the variables that \( \varphi \) exports. Active substitutions are useful because they allow us to map an extended process A to its frame \( \varphi(A) \) by replacing every plain process in A with 0. The frame \( \varphi(A) \) can be viewed as an approximation of A that accounts for the static knowledge \( \varphi \) exposes to its environment, but not \( \varphi \)'s dynamic behavior. The domain of dom(A) is the domain of \( \varphi(A) \). 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 \( \Sigma \) 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 \( \Sigma |- M = N \) for equality within the equational theory \( \Sigma \) and \( \Sigma |- M = N \) for inequality.

**Operational semantics:** The operational semantics is inherited from the applied PI calculus and is defined by structural equivalence (\( =_s \)) and internal reduction (\( \rightarrow_\alpha \)).

Structural equivalence in Fig. 6 (\( =_s \)) is the smallest equivalence relation on extended processes that is closed by \( \alpha \) 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. \( \alpha_{ \text{sub} } \) enables the introduction of an arbitrary active substitution. \( \alpha_{ \text{ext} } \) describes the application of an active substitution to a process that is in contact with it. Rewrite deals with equational rewriting.

Internal reduction (\( \rightarrow \)) 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:
The Text and E_{act} directly depend on the underlying equational theory; using E_{act} sometimes requires that active substitutions in the context be applied first, to yield ground terms M and N.

We write A |a when A can send a message on a, that is, when \( A \rightarrow a \cdot \text{ev}(M).P \) for some evaluation context \( \text{ev}(\_).\_ \) 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 (\( \equiv \)) is the largest symmetric relation \( R \) between closed extended processes with the same domain such that \( A R B \) implies:

- \( PA=0 \quad A \equiv A[0] \)
- \( PA=A \quad A[(B)C] = (A[B])C \)
- \( PA=C \quad A[B = B[A] \)
- \( Refl \quad !P = P \cdot !P \)
- \( N\rightarrow = 0 \quad \mathrm{n} 0 = 0 \)
- \( N\rightarrow = C \quad \forall u v w A = u v w(A) \)
- \( N\rightarrow = PA \quad A \cdot u v B = u v(A[B]) \text{ when } u \in \text{fv}(A) \cup \text{fv}(A) \)
- \( Also \quad \forall x \{M/x\} = 0 \)
- \( Soc \quad \{\text{M}/x\} A = \{\text{M}/x\} A[\{M]/x\] when \( \Sigma - M = N \)

Fig. 6: Structural equivalence

- \( \text{Cases} \quad \bar{w}(x) \cdot P[a(x), Q \rightarrow P]Q \)
- \( \text{True} \quad \text{if } M = M \text{ then } P \text{ else } Q \rightarrow P \)
- \( \text{False} \quad \text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q \)

for any ground terms M and N such that \( \Sigma - M = N \)

Fig. 7: Internal reduction

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 Diffie-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.en.s.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 (Bachev 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

Security protocols: Applied PI calculus + cryptography
Security properties to prove: Security, receipt-freeness, coercion-resistance,

Automatic translator

Horn clauses Derivability queries

Resolution with selection

True Attack

Fig. 8: Mechanized proof tool ProVerif (Blanchet, 2008)
expanded properties including receipt-freeness and coercion-resistance for Backes et al. (2008a) 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 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:
  
  1. for any $t_1, t_2, v$ such that $t = t_1 :: endvote(v) :: t_2$, there exists $id, t', t^*, t^*$ such that :

  (a) $t_1 = t' :: startid(id) :: t^* :: startcorid(id)$ ensures soundness;

  (b) or $t_1 = t' :: startcorid(id) :: t^* :: t^* :: t_2$ ensures soundness

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

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

**Fig. 9: Formal definition of soundness**
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 \( E_{k}^{C_{i},v} \) is an Extractor if and only if:

\[
E_{k}^{C_{i},v} = \text{let } x_{i} = k \text{ in } v_{i}[c_{i}(x) E_{i}, c_{i}(y) E_{i}, C_{i} \text{ if } z \in v \text{ then } [1]]
\]

For some plain processes \( P_{v}, P_{c}, \) and a sequential context \( C \) such that \( c_{i}, c_{i} \notin \text{fa}(P_{i}) \cup \text{fa}(C_{i}) \cup \text{fa}(C) \), \( z \in \text{captured}(C_{i}) \), all inputs and outputs in \( P_{i}, P_{c}, C \) occur on the private channels in \( ni_{i} \), and such channels are never output.

The channels \( c_{i} \) and \( c_{i} \) are the channels shared by the extractor with the coerced voter and the tallying authority, respectively. If the coerced 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:

\[
E_{k}^{C_{i},v} = \text{let } x_{i} = k \text{ in } v_{i}[c_{i}(x) E_{i}, c_{i}(y) E_{i}, C_{i} \text{ if } z \in v \text{ then } [1]]
\]

can tally at most one vote.

- **Formal definition:** An election context \( S \) guarantees coercion-resistance if there exist channels \( c_{i}, c_{i} \) and \( c_{i} \), a sequential process \( V^{ske} \), an Extractor \( E^{C_{i},v}_{ske} \) and an election context \( S_{i} \), 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 \( c_{i} \), shared with Extractor. In condition 2 the left side process contains the voter \( V_{i} \) that is in accordance with the orders of the coerced, running in parallel with the voter \( V_{i} \), casting a vote \( v' \) and the process \( E^{C_{i},v}_{ske}[0] \), that is intuitively equivalent to a voter nullifying her vote. In the right side election process the voter \( V_{i} \), cheats the coerced 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:

\[
E_{k}^{C_{i},v} = \text{let } x_{i} = k \text{ in } v_{i}[c_{i}(x) E_{i}, c_{i}(y) E_{i}, C_{i} \text{ if } z \in v \text{ then } [1]]
\]

tallies the vote the coerced casts on behalf of \( v_{i} \). In condition 3 if the cheated coerced abstains, then the Extractor needs to abstain as well; In condition 4 if the cheated coerced casts a valid vote using the fake registration secrets he received from \( v_{i} \), the Extractor
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.

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\(_i\), representing shares of credentials, for each eligible voter v\(_j\), (j = 1, ..., j). For each c\(_j\), A\(_i\) performs two operations: first, it encrypts c\(_j\) using PK\(_i\) and appropriate secret randomization, signs the resulting ciphertext with SK\(_i\), and publishes it on bulletin board on a row publicly reserved for the shares of credential of voter:

\[ v_j : (E^i(c_j))SK_A \]

SK\(_A\) represents the signature of authority A. Second, A\(_i\) also encrypts c\(_j\) using PK\(_V\) and appropriate secret randomization, without signing it, but attaching to it a designated verifier proof DVP\(_i\) of equality of plaintexts E\(_i^l(c_j)\) and E\(_i^l(c_j)\). 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:

\[ \text{voter}_j : E^i(E^i(c_j), DVP_i) \]

E\(_V^i\) 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\(_i^l(c_j)\) and the corresponding E\(_i^l(c_j)\) that has been signed and published in her reserved area of bulletin board. Upon successful verification, she multiplies together the shares E\(_i^l(c_j)\):

\[ \prod_{j=1}^{l} (E^i(c_j)) = E^i \left( \sum_{j=1}^{l} c_j \right) = E^i(C_j) \]

C\(_j\) is the sum of the various shares of credentials. Voter chooses the ballot shares E\(_l^i(b_j)\) of equality of plaintexts E\(_l^i(b_j)\) and sends E\(_l^i(E^i(C_j + B_j))\) 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 \( \text{pPKdec}(x, PR) \) and encryption algorithm \( \text{pPKenc}(x, PU, r) \). \( \text{pPKdec}(x, PR) \) decrypt the ciphertext \( x \times x \) with private key \( PR \). \( \text{pPKenc}(x, PU, r) \) encrypt the plaintext \( x \) with public key \( PU \) and random number \( r \). The deterministic public key encryption scheme is expressed

\[
\text{Fun } \text{pPKdec}(x, PR) \quad \text{Fun } \text{add}(x, y) \\
\text{Fun } \text{pPKenc}(x, PU, r) \quad \text{Fun } \text{TpPKdec}(x, \cdots, x_i) \\
\text{Fun } \text{PKdec}(x, PR) \quad \text{Fun } \text{TPKdec}(x, \cdots, x_i) \\
\text{Fun } \text{PK}(x) \quad \text{Fun } \text{VK}(x) \\
\text{Fun } \text{PKenc}(x, PU) \quad \text{Fun } \text{verify}(x, PU) \\
\text{Fun } \text{sign}(x, PR) \quad \text{Fun } \text{projection}(x) \\
\text{Fun } \text{dec}(x, PU) \quad \text{Fun } \text{SelfBlinding}(x, r) \\
\text{Fun } \text{checkciphertext}(x, x_i) \quad \text{Fun } \text{TPKdec}(x, \cdots, x_i) \\
\text{Fun } \text{equals}(x, y) \quad \text{Fun } \text{SK}(x) \\
\]

![Fig. 14: Functions](image)

\[
\begin{align*}
\text{equation } & \text{CheckNZDV}p(DV\text{sign}(x, PR), VK, x) = \text{true}. \\
\text{equation } & \text{equals}(x, x) = \text{true}. \\
\text{equation } & \text{dec}(\text{sign}(x, PR)) = x. \\
\text{equation } & \text{verify}(\text{sign}(x, PR), x) = \text{true}. \\
\text{equation } & \text{pPKdec}(\text{SelfBlinding}(\text{pPKenc}(x, PU, r), t)), PR, x) = x. \\
\text{equation } & \text{add}(\text{projection}(x), \text{projection}(x)) = x. \\
\text{equation } & \text{add}(\text{projection}(x), \text{projection}(x)) = x. \\
\text{equation } & \text{TpPKdec}(\text{TPKdec}(x, PU, r), PR, x) = x. \\
\text{equation } & \text{TpPKdec}(\text{TPKdec}(x, PU, r), PR, x) = x. \\
\text{equation } & \text{TPKdec}(\text{TPKdec}(x, PU, r), PR, x) = x. \\
\text{equation } & \text{TPKdec}(\text{TPKdec}(x, PU, r), PR, x) = x. \\
\end{align*}
\]

![Fig. 15: Equational theory](image)
by decryption algorithm PKdec(x, PR) and encryption algorithm PKenc(x, PU). PKdec(x, PR) decrypts the ciphertext x with private key PR. PKenc(x, PU) encrypts 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 verify(x, PU) verify the digital signature x with public key PU. design(x, PU) decrypt the message from the digital signature x with public key PU.

The probabilistic threshold public key share decryption algorithm TPKsubdec(x, Pr, V, K) decrypt the secret share x with private key PR and verification key V. The probabilistic threshold combining algorithm TPKdec(x, Pr, K) recovers x from x. The deterministic threshold public key share decryption algorithm TPKsubdec(x, Pr, V, K) decrypt the secret share x with private key PR and verification key V. The deterministic threshold combining algorithm TPKdec(x, Pr, K) means that recover x from x.

The projection function projection(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. The check function check(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 ciphers 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:

\[ \text{checkciphertext} \left( \text{PKenc}(x_1, PU_1, r_1), \text{PKenc}(x_1, PU_1, r_1) \right) = \text{true} \]

It can verify the two ciphertext, one is the ciphertext generated with the public key PU_1 and random number r_1, the other is the ciphertext generated with the public key PU_2 and random number r_2, are the same plaintext x_1.

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

\[
\text{Public}(ZK_{\alpha_1}(N, M, F)) = N_{\alpha_1}, \quad \rho \in [0, 1]
\]

\[
\text{Formula}(ZK_{\alpha_1}(N, M, F)) = F.
\]

\[
\text{Ver}_{(F, ZK_{\alpha_1}(N, M, F))} = \text{true} \iff \left[ F(N/\bar{\alpha}) \{M/\bar{\beta}\} = \text{true} \right] \land \left[ F \text{ is an } (i, j) - \text{ formula} \right]
\]

\[
F = F \lor \left[ \beta_{\alpha_2} = \text{check}(\alpha_{\alpha_2}, \beta_{\alpha_2}) \right]
\]

\[
ZK_{\alpha_1, \alpha_2}(N_{\alpha_1}, ... , N_{\alpha_2}, M_{\alpha_1}, ... , M_{\alpha_2}, m, V, K, \rho)
\]

\[
\text{CheckNZDV} \left( \text{DVPsign}(m, PR), VK_{\alpha_2}, m \right) = \text{true} \iff \text{equals}(\text{check}(\alpha_{\alpha_2}, \beta_{\alpha_2}), \beta_{\alpha_2}) = \text{true}
\]

Fig. 17: Model of designated verifier proofs
Fig. 18: Main process

\[
\text{voter} 
\triangleq 
\begin{align*}
\text{in(chVR, id); starttid(id); } \\
\text{in(chVR, kencNZDVP); } \\
\text{in(chVR, kencNZDVP);} \\
\text{let NZDVP}_1 = \text{PKdec}[\text{kencNZDVP}, SK(keyV)] \text{ in} \\
\text{let NZDVP}_2 = \text{PKdec}[\text{kencNZDVP}, SK(keyV)] \text{ in} \\
\text{if CheckNZDVPp(DVPsig}(Public_i,(\text{NZDVP}_1),SK(keyV)),VK(keyV),Public_i,(\text{NZDVP}_1)) \text{ then} \\
\text{if CheckNZDVPp(DVPsig}(Public_i,(\text{NZDVP}_2),SK(keyV)),VK(keyV),Public_i,(\text{NZDVP}_2)) \text{ then} \\
\text{if checkciphertext}(Public_i,(\text{NZDVP}_1),decrypt(Public_i,(\text{NZDVP}_1))) = \text{true then} \\
\text{if checkciphertext}(Public_i,(\text{NZDVP}_2),decrypt(Public_i,(\text{NZDVP}_2))) = \text{true then} \\
\text{let cred} = \prod_{i=1}^{r} \text{Public}_i (\text{NZDVP}_1) \text{ in} \\
\text{let vote} = \prod_{i=0}^{r} \text{vendballot}_i \text{ in} \\
\text{let result} = \text{cred} \times \text{vote} \text{ in} \\
\text{new r;} \\
\text{out}(\text{pub, TpPKenc(result, PK(S), r)}); 
\end{align*}
\]

Fig. 19: Voter process

\[
\text{corruptedvoter} 
\triangleq 
\begin{align*}
\text{in(chVR, id); starttid(id); } \\
\text{in(chVR, kencNZDVP); } \\
\text{in(chVR, kencNZDVP);} \\
\text{let NZDVP}_1 = \text{PKdec}[\text{kencNZDVP}, SK(keyV)] \text{ in} \\
\text{let NZDVP}_2 = \text{PKdec}[\text{kencNZDVP}, SK(keyV)] \text{ in} \\
\text{if CheckNZDVPp(DVPsig}(Public_i,(\text{NZDVP}_1),SK(keyV)),VK(keyV),Public_i,(\text{NZDVP}_1)) \text{ then} \\
\text{if CheckNZDVPp(DVPsig}(Public_i,(\text{NZDVP}_2),SK(keyV)),VK(keyV),Public_i,(\text{NZDVP}_2)) \text{ then} \\
\text{out}(\text{pub, (Public}_i,(\text{NZDVP}_1),Public_i,(\text{NZDVP}_2))); 
\end{align*}
\]

Fig. 20: Corrupted voter process

**Processes:** The complete formal model of Acquisiti 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 Acquisiti 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, chRl, and specifies how the processes are combined in parallel. chVR is the private channel between voter and registration authority, chRI, and chRl 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.
Fig. 21: Registration authority process

Fig. 22: Issuer authority process

Fig. 23: Tallying authority process

Voter process is modeled in applied PI calculus in Fig. 19. Using Paillier encryption, each voter gets the shares ciphertext $kV_{\text{encred}}$, and $kV_{\text{encred}}$ from registration authority, then decrypt and get the credentials $\text{vencred}$, $\text{vencred}$, and the designated verifier proof $\text{NZDVP}_1$ and $\text{NZDVP}_2$. After that the voter use $\text{CheckNZDVP}_p$ to verify $\text{NZDVP}_1$ and $\text{NZDVP}_2$. The voter also use $\text{checkciphertext}()$ to verify the equivalence between the encrypted share: $\text{Public}_1(\text{NZDVP}_1)$, $\text{designt}(\text{Public}_2(\text{NZCVP}_1))$ and the one $\text{Public}_3(\text{NZDVP}_2)$, $\text{designt}(\text{Public}_2(\text{NZCVP}_2))$ the voter has received to its message are also provided to itself. The voter also gets the encrypted shares $\text{vencballot'}$, of the ballot, which she has selected from the bulletin board. He multiplies:
Fig. 24: Cheating voter process

```
cheating voter

in (chVR, id);
in (chVR, kencNZDVP);
in (chVR, kencNZDVP);
let NZDVP = PKdec(kencNZDVP, SK (keyV)) in
let NZDVP = PKdec(kencNZDVP, SK (keyV)) in
if CheckNZDVPp(DVPsign(Publicc(NZDVP), SK (keyV)), VK (keyV), Publicc(NZDVP)) then
if CheckNZDVPp(DVPsign(Publicc(NZDVP), SK (keyV)), VK (keyV), Publicc(NZDVP)) then
if checkciphertext(Publicc(NZDVP), design(Publicc(NZDVP))) = true then
if checkciphertext(Publicc(NZDVP), design(Publicc(NZDVP))) = true then
new fakedc, new fakedc;
on (c, (fakedc, fakedc));
on (chve, (fakedc, fakedc));
let cred = \prod_{i=2}^{\infty} Publicc(NZDVP) in
let vote = \prod_{i=2}^{\infty} vencballot;
let result = cred \times vote in
new r;
on (pub, TpPKenc(result, PK(S), r));
```

Fig. 25: Coerced voter process

```
coerced voter

in (chVR, id);
in (chVR, kencNZDVP);
in (chVR, kencNZDVP);
let NZDVP = PKdec(kencNZDVP, SK (keyV)) in
let NZDVP = PKdec(kencNZDVP, SK (keyV)) in
if CheckNZDVPp(DVPsign(Publicc(NZDVP), SK (keyV)), VK (keyV), Publicc(NZDVP)) then
if CheckNZDVPp(DVPsign(Publicc(NZDVP), SK (keyV)), VK (keyV), Publicc(NZDVP)) then
if checkciphertext(Publicc(NZDVP), design(Publicc(NZDVP))) = true then
if checkciphertext(Publicc(NZDVP), design(Publicc(NZDVP))) = true then
out (c, (Publicc(NZDVP), Publicc(NZDVP)));
on (chve, (Publicc(NZDVP), Publicc(NZDVP)));
```

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 kVenced, and kVenced from registration authority, then decrypt and get the credentials venced, venced, 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, and NZDVP, after that, he simply output all their registration secrets venced, and
Fig. 26: Modified tallying authority process

\[ \text{modified tallying authority} \]
\[
\begin{align*}
\text{let } & \text{cencred} = \prod_{i=1}^{n} \text{TpPKdec(projection, (cred), PK(C), r)} \text{ in} \\
\text{let } & \text{bcencred} = \text{SelfBlinding(cencred, PK(C)) in} \\
\text{let } & \text{venceredvote} = \text{TpPKdec(res, SKi(S)) in} \\
\text{out(chTE, venceredvote);} \\
\text{let } & \text{benceredvote} = \text{SelfBlinding(venceredvote, PK(V)) in} \\
\text{let } & \text{cenballot}^i = \prod_{i=1}^{n} \text{TpPKenc(projection, (ballot), PK(C), r)} \text{ in} \\
\text{let } & \text{test} = \text{bcencred} \times \text{cenballot}^i \text{ in} \\
\text{if } & \text{true} = \text{checkciphertext(test, benceredvote)} \text{ then}
\end{align*}
\]

Fig. 27: Abstained voter process

\[ \text{abstained voter} \]
\[
\begin{align*}
\text{in(chVR, id);} \\
\text{in(chVR, kencNZDVP);} \\
\text{in(chVR, kencNZDVP);}
\end{align*}
\]

Fig. 28: Extractor process

\[ \text{extractor} \]
\[
\begin{align*}
\text{in(chVR, id);} \\
\text{in(chVR, kencNZDVP);} \\
\text{in(chVR, kencNZDVP);} \\
\text{new a, new b;} \\
\text{\{in(chVR, (fakecred, fakecred)); out(a, (fakecred, fakecred))\};} \\
\text{in(chTE, venceredvote);} \\
\text{in(a, (fakecred, fakecred));} \\
\text{let } \text{vote}^i = \prod_{i=1}^{n} \text{venceredvote}^i \text{ in} \\
\text{let } \text{cred} = \prod_{i=1}^{n} \text{fakecred}, \text{ in} \\
\text{if } \text{venceredvote} = \text{vote} \times \text{cred} \text{ then} \\
\text{out(b, vote^i)} \\
\text{in(b, z);} \\
\text{if } z \in \text{ballot}^i \text{ then } \}
\end{align*}
\]

vencred\textsubscript{i} 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, and cred. 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\textsubscript{i} and NZDVP\textsubscript{j}.

The issuer authority is modeled in Fig. 22. The issuer authorities get the shares of ballot by projection, (ballot)\textsuperscript{i} and send sign[pPKenc(e, cred, PK(C), r, SK(C))] which is encrypted with a set of Paillier public parameters by the public channel pub.
Tallying authority process is modeled in Fig. 23. After the voting time expires, the tallying authorities get all the ballots on bulletin board posted by allegedly eligible voters and then mixed it by SelfBlinding(cencred, PK(C)). The shares of credentials posted by the registration authorities are also combined and then mixed SelfBlinding(vencredvote, PK(V)). Tallying authorities thus obtain two lists: a list beencred of encrypted, mixed credentials the registration authorities themselves had originally posted on the bulletin board; and a set cencblend of credentials and ballots, posted on the bulletin board by 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 checkcipherpayload(test, beencredvote) 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 fakedcred, and fakecred, 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, (NZDVP, Public, (NZDVP)) and (Public, (NZDVP), Public, (NZDVP))) to the coerced voter.

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 agent 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 mechanized proved 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).
Fig. 31: Function 1

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), eligibility(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 unbreasability 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 $c_v$ shared with Extractor.
Fig. 33: Equation

\[
\begin{align*}
\text{TPKdec} & \left( \text{TPKdec}(x, PK(y)), \text{projection1}(SK(y)), \text{projection1}(VK(y)) \right) = x, \\
\text{TPKdec} & \left( \text{TPKdec}(x, PK(y)), \text{projection2}(SK(y)), \text{projection2}(VK(y)) \right) = x, \\
\text{TPKdec} & \left( \text{TPKdec}(x, PK(y)), \text{projection1}(SK(y)), \text{projection1}(VK(y)) \right) = x, \\
\text{TPKdec} & \left( \text{TPKdec}(x, PK(y)), \text{projection2}(SK(y)), \text{projection2}(VK(y)) \right) = x. \\
\end{align*}
\]

\[\text{zklver} \left( x, \text{zkl}(m, \text{voter}) \right) = \text{TPKdec} \left( \text{TPKdec}(x, PK(V)), \text{projection1}(SK(V)), \text{projection1}(VK(V)) \right).\]

Fig. 34: Soundness-vote chooser

Condition 2 describes the special observational equivalence between:

\[
\begin{align*}
\delta \left[ \kappa_{\text{vote}(v)}(\gamma) \right] & \kappa_{\gamma} \left[ \kappa_{\text{vote}(v)}(\delta) \right] \\
\delta \left[ \kappa_{\text{vote}(v)}(\gamma) \right] & \kappa_{\gamma} \left[ \kappa_{\text{vote}(v)}(\delta) \right]
\end{align*}
\]

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,ct1));
let zkpen=PKdec(ct,SK(voter)) in
if zkver(zkpen)=true then
let (encred1,encred2)=PK(V),m,vk)=public1(zkpen) in
if check(sign(m,voter),vk)=m then
let cred=multi(encred1,encred2) in
in(chvote,vote);event BEGINVOTE(vote,id);
new r1,new r2;
let encvote=multi(TPKenc(projection1(vote),PK(V),r1),
TPKenc(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 corrupted voter=
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 $E(k,v)\mid [9]$, that is intuitively equivalent to a voter nullifying her vote. In:

$$S\left[ V^{\text{smallest}}(z) \mid V_{v'}^{\text{smallest}} \mid E(k,v)_{z} \mid \bar{c}_{\text{vote}}(z) \right]$$

the voter $V_i$ cheats the coherer 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:

$$E(k,v)_{z} \mid \bar{c}_{\text{vote}}(z)$$

tallies the vote the coherer 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:

$$S\left[ V^{\text{smallest}}(z) \mid V_{v'}^{\text{smallest}} \mid E(k,v)_{z} \mid [9] \right]$$

and

$$S\left[ V^{\text{smallest}}(z) \mid V_{v'}^{\text{smallest}} \mid E(k,v)_{z} \mid \bar{c}_{\text{vote}}(z) \right]$$

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, encred1, encred2));
    in (pub, res);
    let encroted=multi[encred1, encred2] in
    let result=PKdec(res, SK(S)) in
    new r1, new r2;
    TpPKenc(projection1(va), PK(C), r1) in
    TpPKenc(projection2(va), PK(C), r2) in
    let encroted=multi[encroted, encroted] in
    let test1=multi(encroted, encroted) in
    let test2=multi(encroted, encroted) in
    if true=checkcipher(test1, result, C, V) then ENDVOTE(va) else
    if true=checkcipher(test2, result, C, V) then ENDVOTE(vb).

Fig. 37: Soundness-tallying authority

let registration_authority=
    in (chVR, (=n1, nonceV));
    in (chRT, (=n1, nonceT));
    new nonce;
    out (chIR, (n1, nonce));
    in (chIR, (=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 (chVI, (=n1, nonceV));
    in (chIR, (=n1, nonceR));
    new id, event NEWID(id);
    out (chVI, (n2, nonceV, id));
    out (chIR, (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 chVI;
new chII;
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\} \{ corrupted voter\} \{ tallying authority\} \{ registration authority\}
\{ issuer authority\} \{ vote chooser\}

Fig. 40: Soundness process

\text{Fig. 41: The result of soundness}

\text{equation zkver} \begin{vmatrix} \text{(cred, sign (m, voter))}, \\
\{ \text{TTPKenc}(\text{cred, PK(V), r}) \} \end{vmatrix} = \text{true.}

\text{Fig. 42: Coercion-resistance-condition2-additional equation}

\text{free pub.com ( \ast public channel\ast )}
\text{private free c, chvote, chTE, chVE, internal1, internal.}
\text{free va, vb}
\text{free n1, n2}
\text{(\ast voter \ast )}
\text{let vote chooser = out (chvote, va) | out (chvote, vb)}

\text{Fig. 43: Coercion-resistance-condition2-vote chooser}
let vote=

in(shVII,id);
new nonce;
out(shVR,(n1,nonce));
in(shVR,(n2,nonce,ct));
let zk=PKdec(ct,SK(voter)) in
if zkver(zk)=true then
let {enccred1,enccred2,=PK(V),m,vk}=public1(zk) in
if check(sign(m,voter),vk)=m then
let cred=multi(enccred1,enccred2) in
in(shvote,vote);
new r1,new r2;
let envote=multi(TpKEnc(projection1(vote),PK(V),r1),
TpKEnc(projection2(vote),PK(V),r2)) in
let ballot=multi(cred,envote) in
let res=PKenc(ballot,PK(S)) in
out(pub,res).

Fig. 44: Coercion-resistance-condition2-voter

let corruptedvoter=

in(shVII,id);
new nonce;
out(shVR,(n1,nonce));
in(shVR,(n2,nonce,ct));
out(pub,ct).

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

let voterreg =

new nonce1;
out(shVR2,(n1,nonce1));
in(shVR2,(n2,nonce1,cred1,cred2,cred3));
out(internal1,(n2,cred1,cred2,cred3)).

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

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

\[ \text{val}
\begin{vmatrix}
\text{let } x = \text{val}
\begin{vmatrix}
\text{let } y = \text{val}
\begin{vmatrix}
\text{let } z = \text{val}
\begin{vmatrix}
\end{vmatrix}
\end{vmatrix}
\end{vmatrix}
\end{vmatrix}
\]

and

\[ \text{val}
\begin{vmatrix}
\text{let } x' = \text{val}
\begin{vmatrix}
\text{let } y' = \text{val}
\begin{vmatrix}
\text{let } z' = \text{val}
\begin{vmatrix}
\end{vmatrix}
\end{vmatrix}
\end{vmatrix}
\end{vmatrix}
\]

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

let votercast =
in(internal,-n2,cred1,cred2,cred3);
in(internal,-n2,cred1,cred2,cred3);
out(chVE,choice([cred1,cred2,cred3],[fakecred,cred1,cred2,cred3]));
new r;
let envote=TPKEnc(vk,PK(V)) in
let res=PKenc([choice1(cred1,cred2),envote],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,encred1,encred2));
  in(pub,res);
let encared=multi(encred1,encred2) in
let result=PKdec(res,SK(S)) in
  new r1,new r2;
let encvoter=multi(TpPKenc(projection1(va),PK(C),r1),
                       TpPKenc(projection2(va),PK(C),r2)) in
let encvoteb=multi(TpPKenc(projection1(vb),PK(C),r1),
                    TpPKenc(projection2(vb),PK(C),r2)) in
let test1=multi(encared,encvoter) in
let test2=multi(encared,encvoteb) 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(chIR.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 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(chII.id);
  out(chII, id);
  out(pub, id).

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

2005) of verification of Condition 3 in ProVerif. The result shows that the observational equivalence between:

\(\text{vc} S^{*}[\{\varepsilon(x)\}_{i\in\tau^{\sigma}(c^{\omega})} \{\varepsilon \}_{i\in\tau^{\sigma}(c^{\omega})}]\)

is need to be proved. Figure 31-33, 57-69 give the inputs in extension of the PI calculus (Abadi and Blanchet,
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:

$$
\text{vcS'} \left[ \Psi_{V_i^{\text{can}}(z)}^{\text{can}}(\chi_{V_i^{\text{can}}}) \bigg\rvert \chi_{V_i^{\text{can}}}^{\text{can}} \right] \sigma_{\text{encT}(z)}
$$

and

$$
\text{vcS'} \left[ \Psi_{V_i^{\text{can}}(z)}^{\text{can}}(\chi_{V_i^{\text{can}}}^{\text{can}}) \bigg\rvert \chi_{V_i^{\text{can}}}^{\text{can}} \right] \sigma_{\text{encT}(\chi_{V_i^{\text{can}}})}
$$

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:

$$
\text{vcS'} \left[ \Psi_{V_i^{\text{can}}(z)}^{\text{can}}(\chi_{V_i^{\text{can}}}^{\text{can}}) \bigg\rvert \chi_{V_i^{\text{can}}}^{\text{can}} \right] \sigma_{\text{encT}(z)}
$$

and

$$
\text{vcS'} \left[ \Psi_{V_i^{\text{can}}(z)}^{\text{can}}(\chi_{V_i^{\text{can}}}^{\text{can}}) \bigg\rvert \chi_{V_i^{\text{can}}}^{\text{can}} \right] \sigma_{\text{encT}(\chi_{V_i^{\text{can}}})}
$$

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
process new C; new V; new S;
new voter;
new chVR;
new chVR1;
new chVR2;
new chVR3;
new chV7;
new chIR;
new chIR1;
new chIR2;
new chIR3;

out(pub.PK(C));
out(pub.PK(V));
out(pub.PK(S));
out(pub.PK(voter));

| voter | corruptedVoter | tallying_authority | registration_authority
|-------|---------------|-------------------|----------------------|
| issuer_authority | voterchooser | modified_tallying_authority
| modified_registration_authority | votercast | voterreg | coercedVoter | extractor

Fig. 55: Coercion-resistance-condition2-process

SELECTING
Termination warning: v_936 \( \rightarrow \) u_936 & attacker2iu_936 & attacker2iu_937, u_936 \( \rightarrow \) had;
SELECTING
Completing...
Termination warning: v_936 \( \rightarrow \) u_936 & attacker2iu_937, u_936 & attacker2iu_932, u_937 \( \rightarrow \) had;
SELECTING
Termination warning: v_936 \( \rightarrow \) u_937 & attacker2iu_936, u_935 & attacker2iu_937, u_935 \( \rightarrow \) had;
SELECTING 0
200 rules inserted. The rule base contains 192 rules, 203 rules in the queue.
800 rules inserted. The rule base contains 388 rules, 146 rules in the queue.
900 rules inserted. The rule base contains 587 rules, 194 rules in the queue.
1000 rules inserted. The rule base contains 787 rules, 199 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, 252 rules in the queue.
1600 rules inserted. The rule base contains 1561 rules, 91 rules in the queue.
2300 rules inserted. The rule base contains 2797 rules, 113 rules in the queue.
RESULT Observational equivalence is true (had not derivable).

El形式：[proverif had proverif.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 voterchooser = out(chvote, va) | out(chvote, vb) .

Fig. 57: Coercion-resistance-condition3-vote chooser
Fig. 58: Coercion-resistance-condition3-voter

```
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 envote = multi (TPKenc (projection1 (vote), PK (V), r1), TPKenc (projection2 (vote), PK (V), r2)) in
    let ballot = multi (cred, envote) in
    let n1 = PKenc (ballot, PK (S)) in
  out (pub, res).
```

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

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

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

```
let voterreg =
  new nonce1;
  out (chVR2, (n1, nonce1));
  in (chVR2, (n2, nonce1, cred1, cred2, cred3));
  out (internal1, (n2, cred1, cred2, cred3)).
```

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

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

```
let tallying_authority =
  new nonce;
out(chRT, (n1, nonce));
in(chRT, (n2, nonce, encr1, encr2));
in(pub, res);
let encr = multi(encr1, encr2) in
let result = PKdec(va, SK(S)) in
new r1, new r2;
let encr = multi(PKenc(projection(1)(va), PK(C), r1), PKenc(projection2(va), PK(C), r2)) in
let encr = multi(PKenc(projection(1)(vb), PK(C), r1), PKenc(projection2(vb), PK(C), r2)) in
let test1 = multi(encr, encr) in
let test2 = multi(encr, encr) in
if true = checkcipher(test1, result, CV) then out(com, va) else
  if true = checkcipher(test2, result, CV) then out(com, vb)
```

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

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

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

```
let issuer_authority =
  new id;
out(chVILid);
out(chIRid);
out(pub, id).
```

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

$$z_{\text{voter}} \equiv b[z_{\text{voter}}]$$
Fig. 69: Coercion-resistance-condition3-process

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

Fig. 71: Coercion-resistance-condition4-vote choose
let voter =
in (chVII.id);
new nonce;
out (chVR,(n1,nonce));
in (chVR,(¬n2,¬nonce,ct));
let zkpk=PKdec(ct,SK (voter)) in
if zkver(zkpk)=true then
let (encr1,encr2,¬PK (V ),m,vk)=public1(zkpk) in
if check(sign(m,voter),vk)=m then
let cred=multi(encr1,encr2) in
in (chvote,vote);
new r1;new r2;
let envote=multi(TpPKenc(projection1(vote),PK (V ),r1),TpPKenc(projection2(vote),PK (V ),r2)) in
let ballot=multi(cred,envote) in
let res=PKenc(ballot,PK (S)) in
out (pub,res).

Fig. 72: Coercion-resistance-condition 4-voter

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

Fig. 73: Coercion-resistance-condition 4-corrupted voter

let voterreg =
new nonce;
out (chVR2,(n1,nonce1));
in (chVR2,(¬n2,¬nonce1,cred1,credE1));
out (internal1,(n2,cred1,credE1)).

Fig. 74: Coercion-resistance-condition 4-voter registration

let coercedvoter =
in (chVII.id);
new nonce;
out (chVR1,(n1,nonce)).
in (chVR1,(¬n2,¬nonce,ct));
let zkpk=PKdec(ct,SK (voter)) in
if zkver(zkpk)=true then
let (encr1,encr2,¬PK (V ),m,vk)=public1(zkpk) in
if check(sign(m,voter),vk)=m then
let cred=encr1 in
new fakcred;
out (internal,(n2,cred,fakcred)).

Fig. 75: Coercion-resistance-condition 4-coerced voter
Fig. 76: Coercion-resistance-condition 4 - voter casting

```plaintext
let voting_cast =
  in(internal,(n2,cred,fairecred));
  in(internal,(n2, cred, cred, cred));
  out(chVE,(fairecred, cred, cred, cred));
  let encvote=TPKenc(va,PK(V)) in
  let res=PKenc((cred, encvote), PK(S)) in
  out(pub,res)
  let encvote=TPKenc(vb,PK(V)) in
  let res=PKenc((fairecred, encvote), PK(S)) in
  out(pub,res).
```

Fig. 77: Coercion-resistance-condition 4 - tallying authority

```plaintext
let tallying_authority =
  new nonce;
  out(chRT,(n1, nonce));
  in(chRT,(n2, nonce, enccred, enccred2));
  in(pub,res);
  let enccred=multi(enccred, enccred2) in
  let result=PKdec(res,SK(S)) in
  new r1; new r2;
  let encvote=multi(TPKenc(projection1(va), PK(C), r1),
                    TPKenc(projection2(va), PK(C), r2)) in
  let encvote=multi(TPKenc(projection1(vb), PK(C), r1),
                    TPKenc(projection2(vb), PK(C), r2)) in
  let test1=multi(enccred, encvote) in
  let test2=multi(enccred, encvote) in
  if true=checkciphertext(test1, result, CV) then out(comm, va) else
  if true=checkciphertext(test2, result, CV) then out(comm, vb).
```

Fig. 78: Coercion-resistance-condition 4 - registration authority

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

Fig. 79: Coercion-resistance-condition 4 - issuer authority

```plaintext
let issuer_authority =
  new id;
  out(chII, id);
  out(chIR, id);
  out(pub, id).
```
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
**Fig. 83:** Coercion-resistance-condition4-process

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

**Fig. 85:** Coercion-resistance-condition5-vote chooser
Fig. 86: Coercion-resistance-condition5-voter

```
let corruptedvote =
  in (chVII,id);
  new nonce;
  out(chVR,(n1,nonce));
  in(chVR,(-n2,-nonce,e,t));
  let skp=PKdec(e,SK(voter)) in
  if skver(skp)=true then
    let (enreed1,reeed2,PK(V),m,vk)=public1(skp) in
    if check(sign(m,voter),vk)=m then
      let cred=multi(enreed1,reeed2) in
      in(chvote,vote);
      new r1,new r2;
      let envote=multi(TpPKenc(projection1(vote),PK(V),r1),TpPKenc(projection2(vote),PK(V),r2)) in
      let ballot=multi(cred,envote) in
      let res=PKenc(ballot,PK(S)) in
      out(pub,res).
```

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

```
let voterchoice =
  new nonce;
  out(chVR,(n1,nonce));
  in(chVR,(-n2,-nonce,e,t));
  let skp=PKdec(e,SK(voter)) in
  if skver(skp)=true then
    let (enreed1,reeed2,PK(V),m,vk)=public1(skp) in
    if check(sign(m,voter),vk)=m then
      let cred=multi(enreed1,reeed2) in
      in(chvote,vote);
      new r1,new r2;
      new falsecred;
      let envote=multi(TpPKenc(projection1(vote),PK(V),r1),TpPKenc(projection2(vote),PK(V),r2)) in
      let ballot=multi(cred,falsecred,envote) in
      let res=PKenc(ballot,PK(S)) in
      out(pub,res).
```

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

The equivalence is true in Fig. 92. Bad is not derivable shows that observational equivalence is true. According to the above analysis, we can find that the Acquisti protocol has the coercion-resistance
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 condition 2 of coercion-resistance is not satisfied. The result is showed in Fig. 93.
CONCLUSION AND FUTURE WORK

Internet voting protocol play an important role in remote voting system. Acquisti protocol is one of the most important remote interents 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 constraints. 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


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.


