情報処理学会論文誌　Vol. 52　No. 9　2549–2561 (Sep. 2011)

電子投票プロトコルに対する無証拠性の定理証明

河辺義信[†1]　真野健[†2]
櫻田英樹[†2]　塚田恭章[†2]

電子投票プロトコルの満たすべき性質に，無証拠性があげられる．これは匿名性の拡張で，「プロトコルを観測して得られる情報に加えて，別の付加的な情報を投票者が外部に与えたとしても，ある投票パターンと別の（結果が同じとなる）投票パターンが区別できない」という性質である．本稿では，Lee らの電子投票プロトコルを題材とし，その無証拠性を定理証明器で形式的に検証する．

Theorem-proving Receipt-freeness of an e-voting Protocol

Yoshinobu Kawabe,[†1] Ken Mano,[†2]
Hideki Sakurada[†2] and Yasuyuki Tsukada[†2]

Receipt-freeness is an important property for an e-voting protocol. This is an extension of anonymity and means that two voting patterns with the same result cannot be distinguishable, even if the voter supplies additional information outside the scope of the protocol. In this paper, we deal with an e-voting protocol by Lee, et al., and we theorem-prove the receipt-freeness of the protocol formally.

1. はじめに

電子投票プロトコルが以下の条件を満たすとき，無証拠性（*receipt-freeness*）を持つという：

（1）　攻撃者は，投票終了後，脅迫などによって「どの候補者に投票したのか」を投票者から聞き出すことができる（聞き出した情報を「レシート」と呼ぶ）．ただし投票者は，せめてもの抵抗として，攻撃者に対し，嘘の情報（嘘のレシート）を渡すことができる．

（2）　投票者は，どの候補者に投票したかを証明できない．つまり，攻撃者が投票結果を検証する際，投票者が正直に白状した場合でも，嘘をついている場合でも，どちらの場合でも検証が成功してしまう．

これは，匿名性を満たし，なおかつ票の買収行為が事実上無意味になることを表す性質である．上記の（2）は，攻撃者の望む候補者に本当に投票されたのか，攻撃者には分からないことを意味する．すなわち，無証拠性を満たす電子投票プロトコルを使う場合，攻撃者から票の開示を要求されたとしても，投票者は嘘の（すなわち，実際に投票した内容とは異なる）投票結果を伝えることができる．攻撃者にとっては，票を買収しても，実際にそのとおり投票したかどうか調べる手段がない．票の買収を防止するという観点から，実際の選挙で電子投票を使う際には，無証拠的なシステムを使うことが重要である．

一般に，電子投票プロトコルが無証拠性を満たすように設計されているかを検証することは，容易ではない．これに対し，ソフトウェア工学の分野では，形式手法（もしくは，数理的技法）と呼ばれる技術が研究されている．形式手法では，プロトコルの動作を論理式やオートマトンなどを用いて記述し，その正しさを形式的に検証する．最近では，この考え方を情報セキュリティ分野に適用し，匿名性などを形式的に証明する研究も現れている（文献 1)，6)，7) など）．

匿名性に関しては，計算機を用いた形式検証[16)] が試みられている．とくに文献 10) などでは，汎用の定理証明器[5)] を用いた匿名性検証法が提案されている．しかし無証拠性については，汎用の定理証明器を用いた自動検証の事例は知られていない．そこで本稿では，電子投票プロトコル[13)] を I/O-オートマトンを用いて形式化し，さらに定理証明器で無証拠性を検証する．本稿の検証では，各候補者が攻撃者以外から最低 1 票は獲得するという仮定のもとに，この 1 票を投じた投票者と他の投票者の間の役割交換（2 人の投票者の投票先を入れ替えた状況を考えても，矛盾が起きないこと）が証明される．

本稿は，以下のように構成される．まず 2 章では，I/O-オートマトンと匿名性検証法を概観する．次に 3 章で，検証の題材である「Lee らの電子投票」の概要について述べ，さらに I/O-オートマトンを用いた電子投票の形式化を示す．4 章では，定理証明器を用いた

†1 愛知工業大学情報科学部情報科学科
Department of Information Science, Aichi Institute of Technology
†2 日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories, NTT Corporation

無証拠性の検証について述べる．無証拠性は，投票者が持つすべての情報を攻撃者が知りうる，という条件下での匿名性と見なせる．そこで本稿では，この条件下での匿名性を，既存の匿名性検証法を利用して形式的に示す．具体的には，Lee らの電子投票が「匿名シミュレーション」と呼ばれる二項関係を持つことを示す．匿名シミュレーションは，状態間の識別不可能性に相当する．5 章では，関連研究などについて述べる．

2. 準　　備

本章では，まず I/O-オートマトンと仕様記述言語について述べ，それから匿名性の形式化と検証法について述べる．

2.1 I/O-オートマトン

I/O-オートマトン[14] は，状態機械に基づく形式モデルである．I/O-オートマトン X は，アクションの集合 $sig(X)$，行為者アクション集合族 $act(X)$，状態の集合 $states(X)$，初期状態の集合 $start(X) \subset states(X)$ および遷移の集合

$$trans(X) \subset states(X) \times sig(X) \times states(X)$$

からなる．アクションには，入力，出力，内部の 3 種類があり，それぞれの集合を $in(X)$，$out(X)$，$int(X)$ と書く（これらは，互いに素とする）．また，入力と出力を外部アクションと呼び，$ext(X) = out(X) \cup in(X)$ とする（本稿では，簡単化のため $in(X) = \emptyset$ となる X のみを考える）．本稿では，$\bigcup_{A \in act(X)} A$ の要素を行為者アクションと呼び，

- $\bigcup_{A \in act(X)} A \subset ext(X)$
- 相異なる A，$A' \in act(X)$ は互いに素

とする．行為者アクションの集合は，匿名性を議論したい参加者それぞれが行う同じ種類のアクションを集めたもので，そのようなアクションが複数種類存在するため，集合族になっている．匿名的なシステムでは，攻撃者は行為者アクションを観測してもその種類しか分からず，具体的に誰が行ったかまでは分からない．

I/O-オートマトン X の遷移 $(s, a, s') \in trans(X)$ を，$s \xrightarrow{a}_X s'$ とも書く．また，a が内部アクションのとき，$s \rightarrow_X s'$ とも書く．関係 $\twoheadrightarrow_X$ を，関係 $\rightarrow_X$ の反射推移閉包とする．任意の $a \in sig(X)$ および s，$s' \in states(X)$ に関して，$s \stackrel{a}{\Longrightarrow}_X s'$ は以下を表す：(i) a が外部アクションのときは，ある s_1，$s_2 \in states(X)$ が存在して $s \twoheadrightarrow_X s_1 \xrightarrow{a}_X s_2 \twoheadrightarrow_X s'$．(ii) a が内部アクションのときは，$s \twoheadrightarrow_X s'$．初期状態からはじまる遷移列 α に関して，α に現れるすべての外部アクションを順に並べたものを α のトレースと呼ぶ．I/O-オートマトン X の全トレースの集合を $traces(X)$ と書く．

I/O-オートマトンの理論では，1 対 1 となるアクションの名前がえを，リネーミングと呼ぶ．これに対し，我々は，I/O-オートマトンに関するアクションの名前がえのうち，1 対 1 とは限らないものを擬リネーミングと呼ぶ．擬リネーミング f に関して，オートマトン X の状態 s でアクション a が発火可能なとき，オートマトン $f(X)$ の状態 s でアクション $f(a)$ が発火できる．とくに本稿では，

- $a \in \bigcup_{A \in act(X)} A$ ならば $f(a) = a$.
- $a \notin \bigcup_{A \in act(X)} A$ ならば $f(a) \notin \bigcup_{A \in act(X)} A$.

を満たす X の擬リネーミング f を X へのシールと呼ぶ．写像 f をシールとするとき，写像 $traces \circ f$ は，シール f によって情報を隠蔽した後にトレース集合を求める操作を表している．すなわち，シール f は，盗聴者がシステムからどのような情報を取り出せるか（また，取り出せないか）を指定する操作と見なせる．

2.2 IOA 言語

I/O-オートマトンは，仕様記述言語 IOA[15] を用いて記述できる．IOA 言語において，各オートマトンは，`signature`-部（アクションの名前と引数の型の宣言），`states`-部（変数名，型，および初期値の宣言），および `transitions`-部（遷移の定義）からなる．さらに，`transitions`-部は，アクション名，`pre`-部および `eff`-部の組からなる．

例 1　10 から 0 までカウントダウンするオートマトンの記述例：

```
automaton counter
  signature
    output count(i: Nat)
  states
    c: Nat := s(s(s(s(s(s(s(s(s(s(0)))))))))))
  transitions
    output count(i)
      pre c = s(i)
      eff c := i
```

ただし，自然数を表すソート `Nat` や各種記号（代入 `:=`，後者関数 `s` など）は，別途，定義されている．オートマトン `counter` は，以下の 3 つの部分を持つ：

(1)　`signature`-部：出力アクション `count(i)`（ただし，`i` のソートは `Nat`）が宣言されている．

(2)　`states`-部：値を保持する変数 `c` が宣言されている．

(3)　`transitions`-部：出力アクション `count(i)` の本体が定義されている．`count(i)` は，

自然数 c が s(i) の形で表されるとき（すなわち，0 でないとき），発火可能である．
発火の際，c を 1 だけ減じて，c に再格納する．

オートマトンの状態は，states-部で定めた変数の持つ値の組として形式化される．また，状態遷移は，pre-部の条件が成立する状態から eff-部を実行した結果の状態への遷移として定義する．

2.3　匿名性の形式化と検証法

本節では，文献 10) の匿名性の形式化を述べる．また，文献 11)，12) の，盗聴者よりも強い攻撃者を考える場合の匿名性の定義について述べる．

2.3.1　攻撃者として盗聴者を考える場合

攻撃者として盗聴者のみを考えるとき，匿名性は，自明に匿名的なシステムを用いて定義される．

定義 1　オートマトン X に関して，次でオートマトン $anonym(X)$ を定める．

$$
\begin{aligned}
&states(anonym(X)) = states(X),\\
&start(anonym(X)) = start(X),\\
&ext(anonym(X)) = ext(X),\\
&int(anonym(X)) = int(X),\\
&act(anonym(X)) = act(X),\\
&trans(anonym(X)) = trans(X) \cup \{(s_1, a, s_2) \mid (s_1, a', s_2) \in trans(X)\\
&\qquad\qquad \wedge A \in act(X) \wedge a' \in A \wedge a \in A\}
\end{aligned}
$$

直感的には，$anonym(X)$ は，次の意味で匿名的である：

ある $someone \in A$（ただし $A \in act(X)$）に関して $s \xrightarrow{someone}_{anonym(X)} s'$ ならば，任意の $everyone \in A$ に関して $s \xrightarrow{everyone}_{anonym(X)} s'$.

よって，$traces(anonym(X)) = traces(X)$ ならば，$anonym(X)$ の匿名性から X の匿名性が導ける．

定義 2（トレース匿名性[10]）　オートマトン X に関して，$traces(anonym(X)) = traces(X)$ のとき，トレース匿名的という．

オートマトン $f(X)$（ただし f は，X へのシール）がトレース匿名的なとき，$(traces \circ f)(X) = (traces \circ f)(anonym(X))$ が成り立つ．つまり，$f(X)$ のトレース匿名性を示すことは，操作 $traces \circ f$ を用いて観測したときのシステム X と匿名的なシステム $anonym(X)$ の識別不可能性を示すことに相当する．

次に，トレース匿名性の検証方法について述べる．

定義 3　オートマトン X および X へのシール f を考える．X の（f に関する）匿名シミュレーション as は，次を満たす $states(X)$ 上の二項関係である：

(1)　任意の $s \in start(X)$ に関して，$as(s, s)$.

(2)　任意の $s_1, s_2, s_1' \in states(X)$ および $a \in sig(X)$ に関して，$as(s_1, s_1')$ かつ $s_1 \xrightarrow{a}_X s_2$ ならば，

(a)　ある $A \in act(X)$ に関して $a \in A$ ならば，すべての $a' \in A$ に関して，ある状態 s_2' が存在して $as(s_2, s_2')$ かつ $s_1' \stackrel{a'}{\Longrightarrow}_X s_2'$.

(b)　$a \notin \bigcup_{A \in act(X)} A$ ならば，ある状態 s_2' およびアクション a' が存在して，$as(s_2, s_2')$，$s_1' \stackrel{a'}{\Longrightarrow}_X s_2'$ かつ $f(a) = f(a')$.

次の定理より，匿名シミュレーションを見つけることで，トレース匿名性を証明できる．

定理 1　オートマトン X および X へのシール f を考える．X の匿名シミュレーション as が存在するならば，$f(X)$ はトレース匿名的である．　□

すなわち，匿名シミュレーションを 1 つ見つければ，ある参加者による任意のトレースに対して，それと区別できない他の参加者によるトレースの存在が保証される．

2.3.2　盗聴者よりも強い攻撃者の場合

攻撃者が盗聴者よりも強い場合，攻撃者の状態や攻撃イベントの発生をモデル化する必要がある．以下では，簡単化のため，シールが恒等関数である場合で説明する．文献 11)，12) では，以下のモデル化が行われている．

定義 4（攻撃部）　4 項組 $(states(X), S_{Atk}, A_{Atk}, T_{Atk})$ で，オートマトン X に対する攻撃部 Atk を定める．ただし，S_{Atk}，A_{Atk} および T_{Atk} は，

$$
\begin{cases}
sig(X) \cap A_{Atk} = \emptyset,\\
T_{Atk} \subseteq \{((s_1, v_1), a, (s_2, v_2)) \mid s_1, s_2 \in states(X), v_1, v_2 \in S_{Atk}, a \in A_{Atk}\}
\end{cases}
$$

を満たすとする．

定義 4 で，S_{Atk} は攻撃部 Atk の内部状態の集合，A_{Atk} はアクションの集合，T_{Atk} は遷移の集合である．

定義 5（プロトコルと攻撃部の合成）　X をオートマトン，X に対する攻撃を $Atk = (states(X), S_{Atk}, A_{Atk}, T_{Atk})$ とするとき，オートマトン (X, Atk) を次のように定める．

$$states((X, Atk)) = \{(s, v) \mid s \in states(X), v \in S_{Atk}\},$$
$$start((X, Atk)) = \{(s, v) \mid s \in start(X), v \in S_{Atk}\},$$
$$ext((X, Atk)) = ext(X) \cup A_{Atk},$$
$$int((X, Atk)) = int(X), \quad act((X, Atk)) = act(X),$$
$$trans((X, Atk)) = \{((s_1, v), a, (s_2, v)) \mid (s_1, a, s_2) \in trans(X), v \in S_{Atk}\} \cup T_{Atk}$$

オートマトン (X, Atk) では，攻撃者 Atk のアクションにより，プロトコル X が振る舞わされる（遷移させられる）ことがある．これにより，盗聴者より強い攻撃が形式化される．

匿名性の定義には，次のオートマトンが用いられる．

定義 6　オートマトン X および X に対する攻撃 $Atk = (states(X), S_{Atk}, A_{Atk}, T_{Atk})$ を考える．また，記号 $start$ および $init$ に関して，$start \notin states((X, Atk))$ かつ $init \notin sig(X) \cup A_{Atk}$ とする．このとき，$\overline{(X, Atk)}$ を，以下で定める．

$$states(\overline{(X, Atk)}) = states((X, Atk)) \cup \{start\},$$
$$start(\overline{(X, Atk)}) = \{start\},$$
$$ext(\overline{(X, Atk)}) = \{(o, v) \mid o \in (ext((X, Atk)) \cup \{init\}) \backslash (\textstyle\bigcup_{A \in act(X)} A),\ v \in S_{Atk}\} \cup (\textstyle\bigcup_{A \in act(X)} A),$$
$$int(\overline{(X, Atk)}) = \{(i, v) \mid i \in int((X, Atk)),\ v \in S_{Atk}\},$$
$$act(\overline{(X, Atk)}) = act(X),$$
$$trans(\overline{(X, Atk)}) = \{(t, (a, v_2), (s_2, v_2)) \mid a \notin \textstyle\bigcup_{A \in act(X)} A, (t, a, (s_2, v_2)) \in trans((X, Atk))\} \cup \{(t_1, a, t_2) \mid A \in act(X), a \in A, (t_1, a, t_2) \in trans((X, Atk))\} \cup \{(start, (init, v), (s, v)) \mid (s, v) \in start((X, Atk))\}$$

一般に，オートマトン $\overline{(X, Atk)}$ のトレースは，

- オートマトン (X, Atk) のイベント列
- 攻撃部 Atk の内部状態の履歴

の両方を表す．そのため，以下のように匿名性を定めることで，トレース集合上で定められた匿名性（定義 2）を，盗聴者よりも強い攻撃者の場合に拡張できる．

定義 7（攻撃 Atk に関する匿名性[11),12)]）　オートマトン X および攻撃部 Atk に関して，$\overline{(X, Atk)}$ がトレース匿名的ならば，X は Atk に対して匿名的という．

次の定理は，定義 7 の匿名性を検証するための手法を与えている．具体的には，まず匿名シミュレーションの存在を示し（攻撃として盗聴のみを考えることに相当する），それから攻撃部のアクション A_{Atk} に関するステップ対応を示すことで，攻撃 Atk に対する X の匿名性が証明される．

定理 2（攻撃者を考慮した場合の匿名性検証法）　オートマトン X および X に対する攻撃者 $Atk = (states(X), S_{Atk}, A_{Atk}, T_{Atk})$ を考える．以下の両方が成り立つとき，オートマトン X は攻撃者 Atk に対して匿名的である：

(1)　X が匿名シミュレーション as を持つ．

(2)　$s_1, s_2 \in states((X, Atk))$ および $a \in A_{Atk}$ に関して，$s_1 \xrightarrow{a}_{(X,Atk)} s_2$ かつ $\overline{as}(s_1, s_1')$ のとき，ある $s_2' \in states((X, Atk))$ が存在して $s_1' \overset{a}{\Longrightarrow}_{(X,Atk)} s_2'$ かつ $\overline{as}(s_2, s_2')$．

ただし $\overline{as} = \{((s_1, v), (s_2, v)) | s_1, s_2 \in states(X), as(s_1, s_2), v \in S_{Atk}\}$ である．　□

3. Lee らの電子投票とその形式化

Lee らの電子投票における参加者は，ある有限な数の投票者（`i`），集計者（`tallier`）および選挙管理人（`adm`）である．ただし `i` は，投票者の識別子の集合を表すソート `ID` の値である．本稿では，攻撃者は投票しないものとする．

3.1　プロトコルの概要

投票プロトコル（**図 1**）は，以下のフェーズを持つ：

セットアップ・登録：プロトコル動作の準備．ここで，各投票者 `i` は選挙管理人に自身の届出を行い，耐タンパランダマイザ（TRR）と呼ばれる，票の再暗号化を行うデバイスを 1 つ受け取る（本稿では，ElGamal 暗号などの再暗号化可能な暗号の利用を前提とする）．TRR の内部のデータは投票者にも分からず，攻撃者に開示されない．また，以下で述べる投票者 `i` とその TRR との間のやりとりは，外部からは観測されない．

投票券の作成：まず投票者 `i` は，集計者の公開鍵 `pub(tallier)` とノンス `r[i]` を使って，投票内容 `cand[i]` を暗号化関数 `enc` で暗号化する．さらに，結果

$$v_i = \texttt{enc(cand[i], pub(tallier), r[i])}$$

を TRR に送る．TRR はノンス `forReEnc[i]` を生成し，再暗号化のための関数 `reEnc` を用いて値 $v_i' = \texttt{reEnc}(v_i, \texttt{forReEnc[i]})$ を得る．さらに TRR は，選挙管理人の秘密鍵 `prv(adm)` で値 $\texttt{reEnc}(v_i, \texttt{forReEnc[i]})$ に対して署名した結果

$$s_i = \texttt{sign(reEnc}(v_i, \texttt{forReEnc[i]}), \texttt{prv(adm))}$$

と DVRP（designated-verifier re-encryption proof）と呼ぶデータ

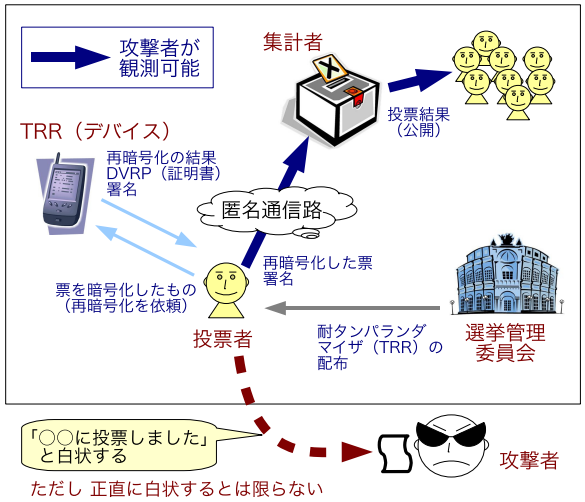

図 1　Lee らの電子投票プロトコル
Fig. 1　Overview of an e-voting protocol.

$dvrp_i$ = `dvrp(`v_i`, reEnc(`v_i`, forReEnc[i]), forReEnc[i], pub(i))`
を計算してから，組 $(v_i', dvrp_i, s_i)$ を投票者 `i` に返送する．DVRP は，再暗号化が正しく行われたことを示すためのデータである．また，関数 `sign` と `dvrp` は，それぞれ署名と DVRP を求める関数である．`reEnc` の第 2 引数と `dvrp` の第 3 引数には，同じノンス `forReEnc[i]` が使われる．

投票券の送出：投票者は，$dvrp_i$ を使って v_i' が正しい再暗号化かを確認し，匿名通信路で集計者に (v_i', s_i) を送る．集計者は，値 (v_i', s_i) を受け付けると，署名の正当性を判定する述語 `checkSign` を用いて選挙管理人の署名 s_i の正当性を確かめる（結果 chk_i = `checkSign(`s_i`, pub(adm)` は，真または偽）．さらに，(v_i', s_i, chk_i) を公開の掲示板 `bbd` に書き出す．

集計：すべての投票者が投票を終えたら，集計者は掲示板 `bbd` に書かれた結果をもとに，集計を行う．最後に，集計結果 `tally(bbd)` が公表される．ただし `tally` は，掲示板 `bbd` の内容を入力し集計結果を出力する関数である．

本稿では，暗号化・復号化などの関数（`enc`，`dec` など）や署名検証などの述語（`checkSign` など）を，次のような等式で形式化する．

```
dec(enc(m, pub(i), m'), prv(i)) = m
reEnc(enc(mcand, pub(i), m1), m2) = enc(mcand, pub(i), newKey(m1, m2))
checkSign(sign(m, prv(i)), pub(i))
checkDVRP(dvrp(m, reEnc(m, m'), m', pub(i)), m, reEnc(m, m'), pub(i))
checkDVRP(dvrp(m1, m2, m3, prv(i)), m1, m2, pub(i))
```

ただし，引数 `i` には投票者名，`mcand` には候補者，また `m`，`m'`，`m1`，`m2`，`m3` には任意の値が入る．暗号化，再暗号化，DVRP はすべて確率的アルゴリズムであり，対応する関数記号 `enc`，`reEnc`，`dvrp` のそれぞれ第 3 引数，第 2 引数，第 3 引数はランダムなビット列に対応する記号である．

3.2　プロトコルの形式化

上記のプロトコルを I/O-オートマトン[14] で記述する．**図 2** は，プロトコルの I/O-オートマトンによる記述 `leeVotingPro` である．セットアップ・登録フェーズの動作は，オートマトン `leeVotingPro` には記述されておらず，トレース上に現れない．すなわち，セットアップ・登録フェーズで公開される情報はないものとして，プロトコルは形式化される．オートマトン `leeVotingPro` の状態は，次に述べる変数の値の組として表現される（変数 `bbd` を除き，各変数は投票者 `i` を添字にとる配列変数である．たとえば，投票者 `i` の投票先やノンスは，それぞれ `cand[i]` や `r[i]` のように書く）：

- `cand`：各投票者の投票先の配列．
- `r`：各投票者が暗号化の際に用いるノンスの配列．
- `forReEnc`：各投票者の TRR が再暗号化のために使うノンスの配列．
- `pc`：各投票者の進捗状態の配列．
  - `pc[i]` がとりうる値は，`start`（投票の開始時），`prepare`（投票者 `i` が TRR に値を送るまで），`wait`（投票者 `i` が TRR から値を受けとるまで），`send`（投票者 `i` は集計者に票を送信中），`blocked`（投票を終えてから開票結果が出るまで），`done`（投票処理の終了時）のどれかである．
- `mes1`：各投票者が TRR に値を渡すためのバッファの配列．
- `mes2`：各投票者が TRR から値を受けとるためのバッファの配列．
- `bbd`：掲示板の内容．
- `ident`：各投票者が（識別子とは別に）固有に持つ値の配列．
- `normIdentPtrn` および `normVotePtrn`：それぞれ変数 `ident` および `cand` の値を初期

```
automaton leeVotingPro
 signature
  output       actorAction(votePtrn, identPtrn),  voter2bbd(i:ID, m:MES),  declare(result:MES)
  internal     voter2trr(i:ID),  trr2voter(i:ID, r:MES)

 states
  cand:  Array[ID, CAND],  r:    Array[ID, MES],  forReEnc:      Array[ID, MES],
  pc:    Array[ID, PC],    mes1: Array[ID, MES],  normIdentPtrn: Array[ID, MES],
  bbd:   Seq[MES],         mes2: Array[ID, MES],  normVotePtrn:  Array[ID, CAND],
  ident: Array[ID, MES]
    such that    pc = constant(start)  /\ bbd = empty
              /\ mes1 = constnat(NULL) /\ mes2 = constant(NULL)

 transitions
  output actorAction(votePtrn:Array[ID, CAND], identPtrn:Array[ID, MES])
   pre (\A i:ID (pc[i] = start))
       /\ (\A i:ID (\E j:ID (votePtrn[i] = normVotePtrn[j] /\ identPtrn[i] = normIdentPtrn[j])))
       /\ (\A j:ID (\E i:ID (votePtrn[i] = normVotePtrn[j] /\ identPtrn[i] = normIdentPtrn[j])))
   eff pc := constant(prepare);
       cand := votePtrn;
       ident := identPtrn

  internal voter2trr(i)
   pre pc[i] = prepare
   eff pc[i] := wait;
       mes1[i] := enc(cand[i], pub(tallier), r[i])

  internal trr2voter(i, r)
   pre pc[i] = wait /\ r = forReEnc[i]
   eff mes2[i] := triple(reEnc(mes1[i], r),
                         dvrp(mes1[i], reEnc(mes1[i], r), r, pub(i)),
                         sign(reEnc(mes1[i], r), prv(adm)));
       pc[i] := send

  output voter2bbd(i, m)
   pre pc[i] = send
       /\ checkDVRP(second(mes2[i]), enc(cand[i], pub(tallier), r[i]), first(mes2[i]), pub(i))
       /\ m = triple(first(mes2[i]), third(mes2[i]), checkSign(third(mes2[i]), pub(adm)))
   eff bbd := m -| bbd;
       pc[i] := blocked

  output declare(result)
   pre (\A i:ID (pc[i] = blocked)) /\ result = tally(bbd)
   eff pc := constant(done)
```

図 2　投票プロトコルの仕様
Fig. 2　Formal specification of an e-voting protocol.

化するために使われる．

上記の変数のいくつかには，初期値が与えられる（図 2 の states-部の such that 以降）．たとえば，配列変数 pc の初期値は constant(start) であり，この初期値は，すべての添字 i について pc[i] が start であることを表す．また，掲示板を表す列 bbd の初期値は空列 empty である．

以下では，各アクションと 3.1 節で示した各フェーズの対応を述べる．

actorAction(votePtrn, identPtrn)：行為者アクションで，各候補者が投票先を決めるために用いられる．すべての投票者の状態が start のとき，アクション actorAction は発火可能である．発火の際，全投票者に関する投票先の分布 votePtrn が求められ，配列変数 cand に格納される．

voter2trr(i)：「投票券の作成」フェーズの前半部分に対応．投票者 i が TRR に値 v_i = enc(cand[i], pub(tallier), r[i]) を渡すまでの動作である．voter2trr(i) は内部アクションであるため，観測されない．

trr2voter(i, r)：「投票券の作成」フェーズの後半部分に対応．投票内容の再暗号化 v_i'，DVRP $dvrp_i$，および署名 s_i を TRR が作成して，投票者に組 $(v_i', dvrp_i, s_i)$ を渡す動作である．trr2voter(i, r) は内部アクションであるため，観測されない．

voter2bbd(i, m)：「投票券の送出」フェーズに対応．出力アクション voter2bbd は引数 i と m を持つため，（匿名通信路を使うにもかかわらず）投票者名 i が観測されてしまうように見える．しかし，本稿では

seal(voter2bbd(i, m)) = voter2bbd(sealDT(m))

を満たすシール seal を導入する（等式の右辺には，i は現れない）．これにより，投票者名 i は観測されないものとして形式化される．

declare(result)：「集計」フェーズに対応．

3.3　攻撃者へのレシートの開示

各投票者は，投票終了後にレシートを作る．ただしレシートは，真の投票結果に基づいて作らなくてもよい．レシートが含む情報は，投票者が真の投票結果をもとにする場合は

1. TRR のデータをもとにした DVRP
2. 投票先の候補者名
3. 投票者が最初の暗号化で使ったノンス
4. TRR が再暗号化したデータ
5. 自身の秘密鍵
6. DVRP の正しさの確認結果（つねに真）

であり，偽りの投票結果をもとにする場合は

1. 自身の秘密鍵を使って偽造した DVRP
2. （偽りの投票先である）候補者名
3. 最初に暗号化する際に使ったノンス
4. TRR が再暗号化したデータ
5. 自身の秘密鍵
6. 偽造 DVRP に対する確認結果（つねに真）

である．上記の情報には，たとえば TRR が選挙管理人の秘密鍵で署名した結果 s_i = `sign(reEnc(`v_i`, forReEnc[i]), prv(adm))` は含まれていない．しかし値 s_i は，オートマトン `leeVotingPro` の `trr2voter(i, r)` アクションによって変数 `mes2[i]` に反映され，さらに `voter2bbd(i, m)` によって，値 `m` の一部として公開される．すなわち攻撃者は，出力アクション `voter2bbd(i, m)` の出現を観測することで s_i を得ている．一般に，攻撃者が得る情報には，

- 外部アクションの出現を観測して得られる情報（トレースに現れる情報）
- レシート上に反映される情報

の 2 種類がある．

図 3 に，投票者の動作を含めた電子投票の仕様 `leeVotingUser` を示す（ただし `leeVotingPro` と共有するアクションや変数などは省略し，投票者の動作部分のみを示す）．`leeVotingUser` は，3 つの変数

`fakecand`	候補者の配列
`r2`	ノンスの配列
`receipt`	レシート格納用の配列

を用いている．`fakecand[i]` の値は，`cand[i]` の値によらず自由に選んだ候補者名である．`fakecand[i]` と `cand[i]` が同じときは投票者 `i` は正直に（すなわち，真の投票結果に基づき）レシートを生成し，異なるときは `fakecand[i]` に基づいてレシートを偽造する．偽のレシートを作る際は，TRR によるノンス（`forReEnc[i]`）が使えないので，何か適当なノンスが必要である．本稿では，値 `r2[i]` をそのノンスとする．攻撃者は，レシート中のノンスが TRR によるノンス（`forReEnc[i]`）なのか投票者がランダムに選んだもの（`r2[i]`）なのかは，判別できないと考えてよい．実際，無証拠性の証明では，

```
sealDT(s.forReEnc[i]) = sealDT(s.r2[i])
```

```
automaton leeVotingUser
 signature
  ＜ leeVotingPro と共通する部分は，省略＞
  internal disclose(i:ID, c:CAND)

 states
  ＜ leeVotingPro と共通する部分は，省略＞
  fakecand:    Array[ID, CAND],
  r2:          Array[ID, MES],
  receipt:     Array[ID, MES] := constant(NULL)

 transitions
  ＜ leeVotingPro と共通する部分は，省略＞
  internal disclose(i, c)
   pre pc[i] = done /\ c = fakecand[i]
   eff if (cand[i] ~= fakecand[i]) then
         receipt[i] := tup(dvrp(enc(c, pub(tallier), r[i]), first(mes2[i]), r2[i], prv(i)),
                           c2m(c),
                           r[i],
                           first(mes2[i]),
                           prv(i),
                           checkDVRP(dvrp(enc(c, pub(tallier), r[i]),
                                          first(mes2[i]), r2[i], prv(i)),
                                     enc(c, pub(tallier), r[i]),
                                     first(mes2[i]),
                                     pub(i)))
       else receipt[i] := tup(second(mes2[i]),
                              c2m(cand[i]),
                              r[i],
                              first(mes2[i]),
                              prv(i),
                              checkDVRP(second(mes2[i]),
                                        enc(cand[i], pub(tallier), r[i]),
                                        first(mes2[i]),
                                        pub(i)))
       fi
```

図 3　投票者を含めた動作仕様
Fig. 3　Voter's action.

という等式を用いている．ここで，`s.`α は状態 `s` での変数 α の値，`sealDT` は攻撃者からデータがどのように見えるかを定める関数である．

投票者 `i` は，自身の秘密鍵 `prv(i)` を使って DVRP を偽造できる．秘密鍵で作った DVRP

```
automaton leeVoting
 signature
  ＜ leeVotingUser と共通する部分は，省略＞
  output checkReceipt(rct:Array[ID, MES])

 states
  ＜ leeVotingUser と共通する部分は，省略＞
  atkpc:       Bool      := true

 transitions
  ＜ leeVotingUser と共通する部分は，省略＞
  output checkReceipt(rct)
   pre atkpc /\ (\A i:ID (receipt[i] ~= NULL)) /\ rct = receipt
   eff atkpc := false
```

図 4　攻撃者を含めた動作仕様
Fig. 4　Adversary's action.

は，偽造であるにもかかわらず検査にパスするという性質を持つ．この性質は，3.1 節で示した checkDVRP-述語の 2 番目の式で形式化されている．偽造 DVRP の生成部分は，オートマトン leeVotingUser のアクション disclose(i, c) で形式化されている．disclose(i, c) の eff-部には条件分岐文があり，レシートを偽造する場合と正直にレシートを作る場合に分けられている．両者を比べると，変数 receipt[i] に格納されるタプルの

- 第 1 要素（投票者が偽造した DVRP か，TRR による本物の DVRP か）
- 第 2 要素（偽の候補者名か，真の投票先か）
- 第 6 要素の一部（第 1，第 2 要素の違いによって，checkDVRP-述語に与える引数が変わる）

が異なる．他の公開情報については，どちらの場合もまったく同じである．

最後に，攻撃者を含めた仕様 leeVoting を**図 4** に示す．図 4 では，レシート格納用の配列変数 receipt の内容を出力するためのアクション checkReceipt(rct) と，投票終了後に 1 回だけ checkReceipt(rct) を起動するために用いる状態変数 atkpc が導入されている．これらのアクションや状態変数は定義 4 の A_{Atk} と S_{Atk} に対応しており，それぞれ，

$A_{Atk} = \{\texttt{checkReceipt(rct)} \mid \texttt{rct}$ は変数 receipt がとりうる任意の値 $\}$

$S_{Atk} = \{\texttt{true}, \texttt{false}\}$

である．本稿の形式化では，すべての投票者がレシートを開示する．

4. 無証拠性の検証

本章では，前章の仕様に対して無証拠性を検証する．投票者が知っている情報をすべて攻撃者が知っていたとしても匿名性が成り立つのが無証拠性である．したがって，投票者の持つ情報を攻撃者に開示するフェーズを付け加えた投票プロトコルを考え，その匿名性として無証拠性を定式化し証明すればよい．すでに 3.3 節で，投票プロトコル（レシートを作成する投票者の動作を含む）についてはオートマトン leeVotingUser として，投票者の情報を攻撃者に開示するフェーズについては，オートマトン leeVoting のアクション checkReceipt(rct) として，それぞれ形式的記述を示している．本章では，まず準備となる条件について整理した後，システム leeVotingUser が匿名シミュレーション関係を持つことを証明する．さらに定理 2 を適用し，攻撃者のアクション checkReceipt(rct) を考慮する場合に定義 7 の意味での匿名性が成り立つことを示す．これにより，上記の「投票者の情報を開示するフェーズを加えた投票プロトコルの匿名性としての無証拠性」が形式的に示される．

4.1 準備：前提条件など

本稿では，偽造レシートを実際に使えるようにするため，「すべての候補者は，攻撃者以外から最低 1 票獲得する」という条件を仮定する．この条件は，

```
(\A s:States[leeVoting] (\A c:CAND (\E i:ID (s.cand[i] = c))))
```

と形式化される．もしこの仮定がないと，たとえば，ある候補者 X がまったく票を獲得していないときに「候補者 X に投票した」と主張するレシートが偽物だと分かってしまう．上記の仮定の下では，攻撃者は，嘘をついた投票者がいることまでは検知できるかもしれない．しかし，各投票者は「自分は嘘をついていない」と言い張れるため，結局，誰が嘘をついているかまでは特定されない．つまり，この条件は，各投票者が自由に嘘をつけることを保証する．

無証拠性の検証の際には，攻撃者からのデータの見え方を定める必要がある．本研究では，関数 sealDT により，それを定める．たとえば，

```
sealDT(reEnc(m, r)) = r                                    (1)
```

は，データ m をノンス r で再暗号化した結果，m の情報が攻撃者から見えなくなることを表している．ある定数 LookLikeNoise を用いて

```
sealDT(reEnc(m, r)) = LookLikeNoise                        (2)
```

とする形式化も考えられるが，本稿では，2 つの再暗号化されたデータに対して，攻撃者は

その同一性は見分けられると考えて，ノンスに置き換えている．等式 (2) を使うと，m の再暗号化 seadDT(reEnc(m, r)) と m’ の再暗号化 sealDT(reEnc(m’, r)) が一致してしまう．等式 (1) を使う場合でも，プロトコル内で r を使った再暗号化が「ただ 1 度だけ」起こる，という前提が必要である．本稿で扱うプロトコルは，その前提を満たしている．

4.2　無証拠性の定理証明

以下に，状態集合 *states*(leeVotingUser) 上の二項関係を示す．

```
as(s, s') <=>
  (   (\A i:ID (s.r2[i] = s'.r2[i]))
   /\ (\A i:ID (s.forReEnc[i] = s'.forReEnc[i]))
   /\ (\A i:ID (s.fakecand[i] = s'.fakecand[i]))
   /\ sealDT(s.bbd) = sealDT(s'.bbd)
   /\ (\A i:ID (s.r[i] = s'.r[i]))
   /\ (\A i:ID (\E j:ID (s.cand[i] = s'.cand[j] /\ s.ident[i] = s'.ident[j])))
   /\ (\A j:ID (\E i:ID (s.cand[i] = s'.cand[j] /\ s.ident[i] = s'.ident[j])))
   /\ (\A i:ID (sealDT(s.receipt[i]) = sealDT(s'.receipt[i])))
   /\ s.normVotePtrn = s'.normVotePtrn
   /\ s.normIdentPtrn = s'.normIdentPtrn
   /\ (   ((\A i:ID (s.pc[i] = start)) <=> (\A i:ID (s'.pc[i] = start)))
       /\ (\A i:ID
             ((s.pc[i] = prepare \/ s.pc[i] = wait \/ s.pc[i] = send)
              <=> (s'.pc[i] = prepare \/ s'.pc[i] = wait \/ s'.pc[i] = send)))
       /\ (\A i:ID (s.pc[i] = blocked <=> s'.pc[i] = blocked))
       /\ ((\A i:ID (s.pc[i] = done)) <=> (\A i:ID (s'.pc[i] = done)))))
```

この二項関係は，

- 状態 s における任意の投票者 i に対して，

  $$(\texttt{s.cand[i]}, \texttt{s.ident[i]}) = (\texttt{s'.cand[j]}, \texttt{s'.ident[j]})$$

  となる投票者 j が s’ に唯一存在する，
- s.fakecand[i] と s’.fakecand[i] は，任意の i に関して同一である，

という条件を含む．これらの条件は，対応する 2 つの状態が，

> 投票内容が互いに役割交換になっており（すなわち開票結果が同じ），かつ攻撃者が観測できるレシートの分布はまったく同じ

となるような異なる 2 つの投票パターンに関するものであることを示している（**図 5**）．

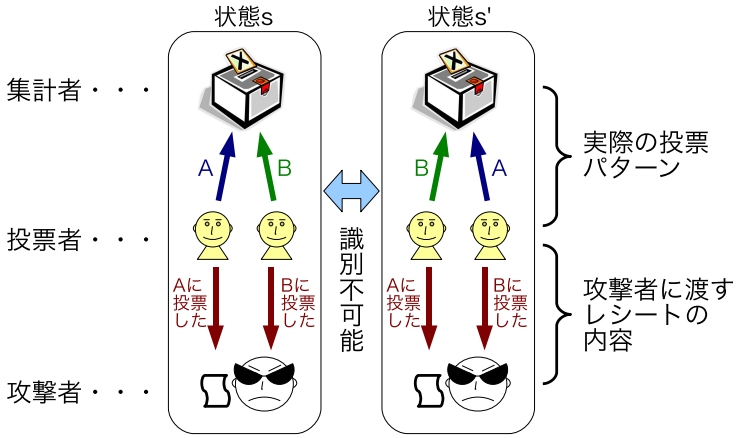

図 5　二項関係 as：開票結果が同じ状態間の識別不可能性
Fig. 5　Indistinguishability of states.

上記の二項関係 as が匿名シミュレーション関係であることを示せば，オートマトン leeVotingUser のトレース匿名性が示される．具体的には，

(1)　初期状態の条件：任意の状態 s について，s が初期状態ならば as(s, s)．
(2)　ステップ対応：leeVotingUser の各アクションが二項関係 as を保存する．

を示せばよい．

まず，as が匿名シミュレーションの初期状態の条件を満たすことを証明する．この証明は，定義 3 の条件 (1) を示すことに対応する．

```
prove
 (\A s:States[leeVoting] (start(s) => as(s, s)))
..
```

ここで，start(s) は，状態 s が初期状態であることを表す述語である．

次にステップ対応の条件，すなわち leeVotingUser の各アクション（ただし actorAction を除く）が二項関係 as を保存することを証明する．この証明は，定義 3 の条件 (2)-(b) を示すことに対応する．以下では，アクション voter2trr(i) のステップ対応に関する 3 つの論理式の証明を示す．これらの論理式では，

- 述語 reachable(s)：状態 s が，ある初期状態から到達可能
- 述語 enabled(s, a)：状態 s において，アクション a が発火可能
- 関数 effect(s, a)：状態 s でアクション a を発火させたときの遷移先

を用いる．

```
% --- voter2trr(i)  ---- 1/3
prove
  (reachable(s1) /\ reachable(s1')
   /\ as(s1, s1')
   /\ enabled(s1, voter2trr(i))
   /\ effect(s1, voter2trr(i)) = s2
   /\ (s1'.pc[i] = prepare))
   => (\E s2':States[leeVoting]
      (\E a':Actions[leeVoting]
          (as(s2, s2')
           /\ a' = voter2trr(i)
           /\ enabled(s1', a')
           /\ effect(s1', a') = s2')))
..

% --- voter2trr(i)  ---- 2/3
prove
  (reachable(s1) /\ reachable(s1')
   /\ as(s1, s1')
   /\ enabled(s1, voter2trr(i))
   /\ effect(s1, voter2trr(i)) = s2
   /\ (s1'.pc[i] = wait \/ s1'.pc[i] = send))
   => as(s2, s1')
..

% --- voter2trr(i)  ---- 3/3
prove
  (reachable(s1) /\ reachable(s1')
   /\ as(s1, s1')
   /\ enabled(s1, voter2trr(i)))
   => (s1'.pc[i] ~= start /\ s1'.pc[i] ~= blocked /\ s1'.pc[i] ~= done)
..
```

最初の 2 つの論理式（1/3 および 2/3）は，as(s1, s1') となる状態 s1' に関する，s1'.pc[i] の値による場合分けである．最後の論理式（3/3）は，s1'.pc[i] の値が start，blocked，done ではないこと（つまり，prepare，wait または send であること）の証明である．

アクション actorAction については，以下の論理式が証明される．この論理式は，行為者アクション（本稿の検証では，各候補者が投票先を決めるために用いる）のためのステップ対応の条件を表しており，「ある投票パターンによる投票が可能であれば，投票結果を変えないような任意の投票パターンが可能である」ことを表す．この論理式を証明することは，定義 3 の条件 (2)-(a) を示すことに対応する．

```
% --- actorAction(votePtrn, identPtrn)
prove
  ((reachable(s1) /\ reachable(s1')
    /\ as(s1, s1')
    /\ enabled(s1, actorAction(votePtrn, identPtrn))
    /\ effect(s1, actorAction(votePtrn, identPtrn)) = s2))
    =>
   (\A a':Actions[leeVoting] (\A vp':Array[ID, CAND] (\A ip':Array[ID, MES]
       ((   a' = actorAction(vp', ip')
         /\ (\A i:ID (\E j:ID (votePtrn[i] = vp'[j] /\ identPtrn[i] = ip'[j])))
         /\ (\A j:ID (\E i:ID (votePtrn[i] = vp'[j] /\ identPtrn[i] = ip'[j]))))
         => (\E s2':States[leeVoting]
                (as(s2, s2')
                 /\ enabled(s1', a')
                 /\ effect(s1', a') = s2'))))))
..
```

以上により，オートマトン leeVotingUser のトレース匿名性が示される．

最後に，文献 11)，12) の結果を用いて，オートマトン leeVotingUser に関するトレース匿名性の結果を，オートマトン leeVoting の匿名性に拡張する．leeVoting は，leeVotingUser に攻撃者のアクション checkReceipt(rct) を加えたものである．定理 2 より，攻撃者のアクション checkReceipt(rct) が二項関係

```
as'(s, s') <=> (as(s, s') /\ (s.atkpc = s'.atkpc))
```

を保存することを示せばよい（二項関係 as' は，定理 2 の二項関係 $\overline{as}$ に対応する）．具体的には，以下の論理式を証明する．

```
% --- checkReceipt(rct)
prove
  (reachable(s1) /\ reachable(s1')
   /\ as'(s1, s1')
   /\ enabled(s1, checkReceipt(rct))
   /\ effect(s1, checkReceipt(rct)) = s2)
   => (\E s2':States[leeVoting]
         (as'(s2, s2')
          /\ seal(checkReceipt(rct)) = seal(checkReceipt(s1'.receipt))
          /\ enabled(s1', checkReceipt(s1'.receipt))
          /\ effect(s1', checkReceipt(s1'.receipt)) = s2'))
..
```

我々は，上記のすべての条件を，Larch 定理証明器[5] を用いて証明した．これにより，投票者が知っている情報をすべて攻撃者が知っていたとしても匿名性が成り立つこと，すなわち電子投票の無証拠性が示された．

定理 3　Lee らの電子投票プロトコルは，無証拠性を満たす．　□

5. 議　　論

本章では，関連研究を述べる．さらに，無証拠性の検証で用いた関数 `sealDT` の正当化について述べる．

5.1 関 連 研 究

Jonker らは，無証拠性を

> プロトコルを観測して得られる情報に加えて，別の付加的な情報を投票者が外部に与えたとしても，ある投票パターンと別の（内訳の同じ）投票パターンが区別できないこと

と考え，知識論理を用いた匿名性の形式化に基づく無証拠性の定義を示した[9]．一方，文献 11)，12) では，盗聴よりも強い攻撃を対象に，I/O-オートマトンを用いた匿名性の形式化と検証手法が提案されている．本稿は，Jonker らと同様の考え方に基づき，文献 11)，12) の I/O-オートマトンによる形式化のもとで無証拠性を匿名性として扱った．さらに，I/O-オートマトンに基づく匿名性の定理証明手法を，無証拠性にも適用した．

本稿では，票の開示を投票終了後に限定する．文献 8) では，攻撃者への情報の渡し方として，

(1)　classic-rf（receipt-freeness）：情報を投票終了後に渡す，
(2)　start-rf：情報を最初に渡す，
(3)　rf-share：メッセージを送出するごとに情報を渡す，
(4)　rf-witness：攻撃者が投票者に対して，プロトコルで使う値を指示する，
(5)　rf-full：(3) と (4) の両方を行う，

という分類をしている．本稿では，最も単純な (1) を扱った．一方，Delaune らは文献 3) で，Lee らの電子投票プロトコルを applied π-計算[1] で形式化し，(2) 以降の場合を扱っている．また，文献 3) の検証は手証明で行われているが，これに対し，本稿の検証は定理証明器により機械化されているという違いがある．

5.2 関数 `sealDT` の正当化について

Abadi らは，対称鍵暗号系に対して，記号的に定義されたメッセージの識別不可能性を与え，さらに，これが計算論的意味において健全であることを示した[2]．計算論的な手法では，データはビット列で表現され，暗号化や鍵生成などの操作は確率的多項式時間アルゴリズムとして扱われる．一方，記号的な手法ではデータは記号として表現され，暗号化などは論理的・代数的に扱われる．

本稿では，記号的な手法に基づいて暗号化関数やプロトコルの動作を記述し，検証している．さらに，攻撃者からのデータの見え方を，関数 `sealDT` を用いて定めている．たとえば，4.1 節で述べた等式

```
sealDT(reEnc(m, r)) = r
```

のほか，DVRP に関しては，等式

```
    sealDT(dvrp(m1, m2, m3, pub(i)))
=   sealDT(dvrp(m1, m2, m3, prv(i)))
=   sealedDVRP(sealDT(m1), sealDT(m2), sealDT(m3), pub(i))
```

を導入して無証拠性の検証に利用している．上記の等式は妥当なものであると考えるが，正当化は必要である．前者の等式は，再暗号化 `reEnc` が乱数性を保存するという計算論的仮定を，等式の形で記号的に表現したものである．また後者は，再暗号化に用いた乱数が分からないとき，公開鍵を用いた DVRP と，秘密鍵を用いて偽造した DVRP が識別不可能であるという計算論的仮定を，記号的に表現したものである．それら計算論的仮定は妥当なも

のである．また，記号的表現も自然なものだが，その正当性の厳密な証明は必要である．

正当性の証明のためには，本稿で用いた再暗号化に対して，Abadi-Rogaway 流の記号的な識別不可能性を定式化することが重要だと考えている．具体的には，妥当な仮定のもとに，

(1)　我々の同値関係に基づく `sealDT` の定義が，Abadi-Rogaway 流の項の構成に基づく識別不可能性の定義と互換性があること，および

(2)　そのような識別不可能性が計算論的に健全であること

を，証明できると考えている．実際の証明は，今後の課題である．

6. むすび

本稿では，Lee らの電子投票プロトコルを形式化し，定理証明器を用いて無証拠性を検証した．具体的には，各候補者が最低 1 票獲得するという仮定のもとに，この 1 票を投じた投票者と他の投票者の間の役割交換を証明した．

今後の課題としては，票の開示を投票終了後に限定しない場合を扱うことと，通信路を表すオートマトンを導入した，より具体的な仕様を使っての無証拠性の検証があげられる．また，最近では，電子投票だけでなく，インターネットオークションプロトコルにおける無証拠性を手証明で示した研究[4] も現れている．こうした手証明による検証を計算機で行うことも，今後の課題にあげられる．

謝辞　本研究へのご支援をいただいた，龍谷大学外村佳伸先生，NTT コミュニケーション科学基礎研究所中岩浩巳部長，真鍋義文主幹研究員に感謝いたします．また本研究の一部は，科研費若手研究（B）（課題番号 23700024）の助成によるものです．ここに謝意を表します．

参考文献

1) Abadi, M. and Fournet, C.: Private authentication, *Theoretical Computer Science*, Vol.322, pp.427–476 (2004).
2) Abadi, M. and Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption), *Proc. TCS '00*, LNCS 1872, pp.3–22 (2000).
3) Delaune, S., Kremer, S. and Ryan, M.F.: Coercion-resistance and receipt-freeness in electronic voting, *Proc. IEEE CSFW '06*, pp.28–39 (2006).
4) Dong, N., Jonker, H. and Pang, J.: Analysis of a receipt-free auction protocol in the applied pi-calculus, *Proc. 7th International Workshop on Formal Aspects of Security and Trust* (*FAST 2010*), LNCS 6561, pp.223–238, Springer-Verlag (2011).
5) Garland, S.J., Guttag, J.V. and Horning, J.J.: An overview of Larch, LNCS 693, pp.329–348, Springer-Verlag (1993).
6) Halpern, J.Y. and O'Neill, K.R.: Anonymity and information hiding in multiagent systems, *Journal of Computer Security*, Vol.13, No.3, pp.483–514 (2005).
7) Hughes, D. and Shmatikov, V.: Information hiding, anonymity and privacy: A modular approach, *Journal of Computer Security*, Vol.12, No.1, pp.3–36 (2004).
8) Jonker, H., Pang, J. and Mauw, S.: A formal framework for quantifying voter-controlled privacy, *Journal of Algorithms in Cognition, Informatics and Logic*, Vol.64, No.2-3, pp.89–105 (2009).
9) Jonker, H.L. and Pieters, W.: Receipt-freeness as a special case of anonymity in epistemic logic, *Proc. IAVoSS Workshop On Trustworthy Elections* (*WOTE 2006*), pp.29–30 (2006).
10) Kawabe, Y., Mano, K., Sakurada, H. and Tsukada, Y.: Theorem-proving anonymity of infinite-state systems, *Information Processing Letters*, Vol.101, No.1, pp.46–51 (2007).
11) 河辺義信，真野　健，櫻田英樹，塚田恭章：能動的な攻撃者が存在するシステムに対する匿名性の検証について，2008 年暗号と情報セキュリティシンポジウム (2008).
12) Kawabe, Y. and Sakurada, H.: An adversary model for simulation-based anonymity proof, *IEICE Trans.*, Vol.E91-A, No.4, pp.1112–1120 (2008).
13) Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J. and Yoo, S.: Providing receipt-freeness in mixnet-based voting protocols, *Proc. 6th International Conference on Information Security and Cryptology* (*ICISC 2003*), LNCS 2971, pp.245–258, Springer-Verlag (2004).
14) Lynch, N.A.: *Distributed algorithms*, Morgan Kaufmann Publishers (1996).
15) Soegaard-Andersen, J.F., Garland, S.J., Guttag, J.V., Lynch, N.A. and Pogosyants, A.: Computer-assisted simulation proofs, *Proc. CAV '93*, LNCS 697, pp.305–319, Springer-Verlag (1993).
16) Schneider, S., Sidiropoulos, A.: CSP and anonymity, *Proc. ESORICS '96*, LNCS 1146, pp.198–218, Springer-Verlag (1996).

(平成 22 年 11 月 30 日受付)
(平成 23 年 6 月 3 日採録)

河辺　義信（正会員）

昭和 47 年生．平成 9 年名古屋工業大学大学院工学研究科博士前期課程電気情報工学専攻修了．同年日本電信電話株式会社入社．NTT コミュニケーション科学基礎研究所にて，項書き換え系，ネットワークプログラミング言語，形式手法の研究に従事．平成 14 年 MIT 計算機科学研究所客員研究員．平成 20 年より愛知工業大学准教授．定理証明系を用いたシステム検証，情報流通セキュリティ分野への形式手法の適用に関する研究に従事．博士（工学）．平成 17 年コンピュータセキュリティシンポジウム優秀論文賞．ACM，電子情報通信学会，日本ソフトウェア科学会各会員．

真野　　健（正会員）

昭和 40 年生．平成元年名古屋大学大学院工学研究科情報工学専攻修士課程修了．同年日本電信電話株式会社入社．現在，NTT コミュニケーション科学基礎研究所主任研究員．項書き換え系，プロセス計算，セキュリティのフォーマルメソッドに関する研究に従事．電子情報通信学会，日本ソフトウェア科学会各会員．

櫻田　英樹（正会員）

昭和 49 年生．平成 11 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年日本電信電話株式会社入社．現在，NTT コミュニケーション科学基礎研究所研究主任．通信プロトコルのセキュリティのフォーマル検証に関する研究に従事．電子情報通信学会，日本ソフトウェア科学会，応用数理学会各会員．

塚田　恭章（正会員）

平成 2 年東京工業大学大学院理工学研究科情報科学専攻修士課程修了．同年日本電信電話株式会社入社．現在，コミュニケーション科学基礎研究所主幹研究員．数理的技法による情報セキュリティの研究に従事．博士（工学）．日本ソフトウェア科学会，日本応用数理学会，ACM 各会員．