電子投票プロトコルの匿名性検証のための関数部分知識モデル

Partial Knowledge of Cryptographic Functions for Verifying the Anonymity of E-Voting Protocols

川本 裕輔 † 真野 健 †† 櫻田英樹 †† 萩谷 昌己 † ††
Yuusuke KAWAMOTO Ken MANO Hideki SAKURADA Masami HAGIYA

† 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, University of Tokyo
{y_kwmt,hagiya}@is.s.u-tokyo.ac.jp
††NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories, NTT Corporation
{mano,sakurada}@theory.brl.ntt.co.jp

This paper proposes a method for formalizing and analyzing a *partial knowledge of cryptographic functions* by using a temporal epistemic logic within a framework of multi-agent systems. The key to expressing the agents' partial knowledge of publicly available cryptographic functions lies in the distinction between the functions available to the agents and the knowledge of the functions acquired by the agents. We consider that agents gradually acquire a partial knowledge of cryptographic functions by performing actions in a multi-agent system model where the interpretation of the cryptographic functions may vary from possible world to possible world. In such a new model, we present a way of verifying the anonymity of security protocols, and apply it to a simple electronic voting protocol.

1 Introduction

Formal theories of knowledge based on multi-agent systems have proved quite useful for formalizing and verifying the anonymity of security protocols. Definitions of anonymity are formalized in terms of knowledge by using an epistemic logic in [9], and the semantic characterization of anonymity in a framework of multi-agent systems is discussed in [4]. In such a formalization of anonymity, the anonymity of Chaum's Dining Cryptographers protocol, a security protocol for anonymous broadcast, is verified in [10]. Although these frameworks are effective for roughly discussing anonymity, they are insufficient to argue the knowledge of cryptographic functions. By introducing the notion of algorithmic knowledge [7], the knowledge of cryptographic functions can be expressed as the local algorithms possessed by agents who use them. However, this formalization of an agent's knowledge of cryptographic functions is not suitable for expressing the partial nature of the knowledge of functions.

This paper proposes a novel method for formalizing a *partial knowledge of cryptographic functions* by using a temporal epistemic logic within a framework of multi-agent systems. We consider the process whereby agents gradually acquire the partial knowledge of cryptographic functions by performing actions. We assume that agents do not have any knowledge of cryptographic functions at the beginning, and that agents' knowledge of cryptographic functions is determined by their own cryptographic actions corresponding to the functions. To formalize secrecy and anonymity of messages in this framework, we take the interpretation of cryptographic functions as varying from possible world to possible world. For example, if a plaintext encrypted in a message is secret, we consider that one plaintext is encrypted in the message at one possible world, and that another plaintext is encrypted at another possible world.

In this model, we introduce the concept of *key-independence*, and define the independence of the execution of the security protocols containing cryptographic operations. By employing this concept of independence and the partial knowledge of cryptographic functions, we obtain a method of verifying the anonymity of security protocols.

We have organized the rest of the paper as follows. Section 2 introduces a model based on multi-agent systems and a partial knowledge of cryptographic functions. Section 3 describes syntax and semantics for a protocol specification. Section 4 presents a method of verifying the anonymity of security protocols in our model, and apply this method to a simple electronic voting protocol. Sec-

tion 5 discusses the limitations of our method and the possibilities of extending our framework.

2 Knowledge in Multi-Agent Systems

In this section, we present a model based on the framework of [8] for describing multi-agent systems. First, we provide a brief review of the framework. Then, we specify the details of our model such as the data domain, actions, and cryptographic functions. Finally, we explain the behavior of the model.

2.1 Overview of Multi-Agent Systems

A *multi-agent system* consists of many possible worlds where each agent is in some *local state*. We assume that an agent's local state contains all the information accessible to the agent. A tuple of the local states of all agents is called a *global state*. A *run* is defined as a function from the time domain, ranging over the natural numbers, to the set of global states. Intuitively, a run expresses a sequence of global states in chronological order. A *point*, or possible world, is a pair (r, m) consisting of a run r and time m. The global state at a point (r, m) is denoted by $r(m)$, and the local state of an agent X at a point (r, m) is denoted by $r_X(m)$. The equivalence of two local states s and s' is written as $s = s'$. We say that two points (r, m) and (r', m') are *indistinguishable* to an agent X (notation: $(r, m) \sim_X (r', m')$), if X has the equivalent local state at (r, m) and (r', m'), that is, $r_X(m) = r'_X(m')$. Note that this indistinguishability relation $\sim_X$ is an equivalence relation.

The knowledge of agents in multi-agent systems can be defined by using the indistinguishability of global states. Let $\mathcal{R}$ be a *system*, that is, a set of runs, X be an agent, and $\mathcal{K}_X(r, m)$ be the set:

$$\mathcal{K}_X(r, m) = \left\{ (r', m') \;\middle|\; \begin{array}{l} r' \in \mathcal{R},\ m' \in \mathcal{N}, \\ r'_X(m') = r_X(m) \end{array} \right\}.$$

Intuitively, X knows a fact ϕ at (r, m) if ϕ is true at all the points X thinks possible at (r, m), that is, in all the possible worlds of $\mathcal{K}_X(r, m)$. It should be noted that agents' knowledge is determined solely by their local states.

To provide a formal definition of knowledge, we need to undertake some preparations. Suppose Φ is a set of atomic formulas. The language in our framework is built up using each atomic formula p in Φ, Boolean connectives, an unary epistemic operator $\mathbf{K}_X$ for each agent X, and unary temporal operators $\bigcirc$, $\bigcirc^{-1}$, and $\Box$. The formula of this language is given by the rule:

$$\phi ::= p \,|\, \neg\phi \,|\, \phi \wedge \phi \,|\, \mathbf{K}_X\phi \,|\, \bigcirc \phi \,|\, \bigcirc^{-1} \phi \,|\, \Box\phi.$$

$\mathbf{P}_X$ is the dual operator for $\mathbf{K}_X$, which is defined by $\mathbf{P}_X\phi := \neg\mathbf{K}_X\neg\phi$. Repetitions of temporal operators are abbreviated as

$$\bigcirc^i\phi := \underbrace{\bigcirc \cdots \bigcirc}_{i}\phi \text{ and } \bigcirc^{-i}\phi := \underbrace{\bigcirc^{-1} \cdots \bigcirc^{-1}}_{i}\phi.$$

We also use the following abbreviations for disjunction and implication: $\phi \vee \psi := \neg(\neg\phi \wedge \neg\psi)$ and $\phi \rightarrow \psi := \neg\phi \vee \psi$. An *interpreted system* $\mathcal{I}$ is a pair $(\mathcal{R}, \pi)$ consisting of a system $\mathcal{R}$ and an interpretation π, which is a truth assignment to each atomic formula in Φ at each point. The satisfaction of a formula ϕ is inductively defined as follows:

- $(\mathcal{I}, r, m) \models p$ if and only if $(\pi(r, m))(p) =$ **true**,
- $(\mathcal{I}, r, m) \models \neg\phi$ if and only if not $(\mathcal{I}, r, m) \models \phi$,
- $(\mathcal{I}, r, m) \models \phi \wedge \psi$ if and only if $(\mathcal{I}, r, m) \models \phi$ and $(\mathcal{I}, r, m) \models \psi$,
- $(\mathcal{I}, r, m) \models \mathbf{K}_X\phi$ if and only if $(\mathcal{I}, r', m') \models \phi$ for any $(r', m') \in \mathcal{K}_X(r, m)$,
- $(\mathcal{I}, r, m) \models \bigcirc\phi$ if and only if $(\mathcal{I}, r, m+1) \models \phi$,
- $(\mathcal{I}, r, m) \models \bigcirc^{-1}\phi$ if and only if $(\mathcal{I}, r, m-1) \models \phi$ and $m \geq 1$,
- $(\mathcal{I}, r, m) \models \Box\phi$ if and only if $(\mathcal{I}, r, m') \models \phi$ for any $m' \geq m$.

2.2 Actions and Cryptography

Any agent can perform a set of *actions* between any successive points. Actions are specified by the data used to perform the actions and the action types. To clarify the data employed in our model, we define the *data domain* $\mathcal{D}$ as a sufficiently large set of data including data representing the names of agents, action types, and available cryptographic functions. We assume that $\mathcal{D}$ is identical in all possible worlds. The types of actions in the model are restricted to communication actions and internal actions.

Communication actions are actions whereby a message is sent from one agent to another agent through a communication channel. Our model is restricted to the case where messages are neither lost nor delayed, and are always received by some agent. We consider two types of communication channels: *normal communication channels* and

anonymous communication channels. The receivers of the messages sent through a normal communication channel know exactly who are the senders. On the other hand, communications through an anonymous communication channel do not leak any information about the senders to any agent other than the senders themselves. Anonymous communication channels are often used in electronic voting protocols [2].

Internal actions in this model are restricted to encryption actions and decryption actions in public-key cryptography. We sometimes refer to these internal actions as *cryptographic actions*. Any cryptographic scheme used in cryptographic actions is represented by a pair consisting of an encryption function and a decryption function from $\mathcal{D} \times \mathcal{D}$ to $\mathcal{D}$. The encryption/decryption functions are associated with each run. In this cryptographic scheme, encrypting/decrypting a message is easy for agents who possess the encryption/decryption key, while agents who do not possess the encryption/decryption key cannot access the contents of a message without requiring a great deal of computation power, because the data domain $\mathcal{D}$ is sufficiently large. In the rest of the paper, cryptographic schemes are limited to $(\sigma_{enc}, \sigma_{dec})$ for simplicity, and all agents can perform *encryption actions* using σ_{enc} and *decryption actions* using σ_{dec}. Note that our model can also deal with random encryption using a nonce by introducing a constant denoting a nonce. However, this paper does not consider random encryption actions for simplicity. Let d be a message, e be an encrypted message, and k_{pub} and k_{sec} be the corresponding public and secret keys. The encryption function σ_{enc} and the decryption function σ_{dec} must satisfy five constraints:

- $\sigma_{dec}(\sigma_{enc}(d, k_{pub}), k_{sec}) = d$
- $\sigma_{dec}(\sigma_{enc}(d, k_{sec}), k_{pub}) = d$
- $\sigma_{enc}(\sigma_{dec}(e, k_{pub}), k_{sec}) = e$
- $\sigma_{enc}(\sigma_{dec}(e, k_{sec}), k_{pub}) = e$
- these two functions are collision-free, that is, they map distinct pairs consisting of a message and a key to distinct encrypted/decrypted messages.

In these constraints, we assume that the cryptographic functions denoted by the same function symbols may have a different interpretation in different runs. The interpretation of cryptographic function symbols is mentioned in a later section.

2.3 Local States

Agents observe actions between any successive points. An action observed between points $(r, m-1)$ and (r, m) from an agent X's viewpoint is referred to as X's *observed action* between the two points. Note that observed actions are not always the same as actually performed actions. An agent X's *action observation* between points $(r, m-1)$ and (r, m) is the set of all of X's observed actions between the two points.

All of the actions any agent observes are recorded in the local state of the agent. The local state of any agent X is a triple of the form:

$$\langle history, data, func \rangle,$$

where *history* is the sequence of sets of actions that X has observed at each time, *data* is the set of data that X knows, and $func$ is all the cryptographic function information known to X. Each of the above items of a local state s is sometimes referred to as $s.history$, $s.data$, and $s.func$, respectively. Two local states s and s' are *equivalent* if s and s' are constructed from the same items. We assume that the *history* and the $func$ of any agent are empty at time 0, which implies each agent has no initial knowledge of cryptographic functions.

Hereafter we consider only the case multi-agent systems are *synchronous perfect recall*. A multi-agent system is *synchronous* if actions take place between points. All agents have access to a shared clock, distinguish the present from the future, and perform actions in synchrony. Agents have *perfect recall* if no agent loses or forgets previously acquired information. In a system with perfect recall, $r_X(m).history$ is the tuple obtained by appending X's action observation between $(r, m-1)$ and (r, m) to $r_X(m-1).history$. Thus, the local state of any agent encodes everything that has happened from the agent's point of view.

To discuss a situation where agents try to overcome secrecy or anonymity, we introduce the two types of agents: normal agents and intruder agents. *Normal agents* can perform all kinds of actions described above, and can observe only their own actions, including communication actions by which they receive messages. On the other hand, *intruder agents* are allowed to perform only internal actions, and to observe all their own actions and all the communication actions performed by any agent. Note that intruder agents cannot observe any internal action of any other agent.

We review the model's behavior for each kind of action. In the following description, any intruder agent is denoted by I, and an action is represented by a quadruple whose first element expresses its action type.

When an agent X sends a message d to another agent Y through a normal communication channel between points $(r, m-1)$ and (r, m), X, Y, and I observe the same action $\langle send_{norm}, d, X, Y\rangle$, where $send_{norm}$ is a semantic entity representing the action type. However, with communications through an anonymous communication channel, observed actions differ between the sender and the receiver, because anonymous communication channels do not leak any information about the senders to the receivers. When an agent X sends a message d to another agent Y through an anonymous communication channel between the points $(r, m-1)$ and (r, m), X observes $\langle send_{ano}, d, X, Y\rangle$, and Y and I observe $\langle send_{ano}, d, \perp, Y\rangle$, in which the symbol $\perp$ is used to represent the lack of information about the sender X. The message d sent in each communication action should be possessed by the sender X, i.e., should belong to $r_X(m-1).data$, or should be a tuple composed of data in $r_X(m-1).data$. When the message d is received by the receiver Y, it is added to $r_Y(m).data$ and $r_I(m).data$. If the received message is a tuple, it is decomposed and its components are added to $r_Y(m).data$ and $r_I(m).data$. The $func$ item of the local state of any agent is unchanged by communication actions.

In contrast to communication actions, internal actions extend the $func$ item of local states. Let ϵ and δ be the data in $\mathcal{D}$ expressing the names of the cryptographic operations corresponding to the encryption function σ_{enc} and the decryption function σ_{dec}. Let k_{pub} be a public key, and k_{sec} be the secret key corresponding to k_{pub}. If an agent X encrypts a message d by using σ_{enc} and k_{pub} and obtains the value e of $\sigma_{enc}(d, k_{pub})$ between points $(r, m-1)$ and (r, m), X observes the internal action $\langle encrypt, \langle d, k_{pub}\rangle, e, X\rangle$ and adds the partial knowledge of cryptographic functions $\langle \epsilon, \langle d, k_{pub}\rangle, e\rangle$ to $r_X(m).func$. As regards X's decryption actions with σ_{dec} and k_{sec}, X observes $\langle decrypt, \langle e, k_{sec}\rangle, d, X\rangle$ and adds $\langle \delta, \langle e, k_{sec}\rangle, d\rangle$ to $r_X(m).func$. When the message e or d is obtained by the encryption or the decryption, it is added to $r_X(m).data$. If the message is a tuple, it is decomposed and its components are added to $r_X(m).data$. The unencrypted message d or undecrypted message e and the key k_{pub} or k_{sec} should be possessed by X, i.e., should belong to $r_X(m-1).data$, or should be tuples composed of data in $r_X(m-1).data$. From this assumption, our model does not treat cryptographic actions using random data, and thus does not allow intruder agents to perform a guessing attack.

When no actions are observed from an agent's point of view, the agent's action observation is the empty set, and the $data$ and the $func$ of the agent remain completely unchanged.

2.4 Knowledge of Cryptographic Functions

Formalizing the secrecy of cryptography is necessary for discussing the anonymity of security protocols. To express the secrecy of public-key cryptography by using a temporal epistemic logic, it is natural to consider a model where the interpretation of cryptographic function symbols may vary from run to run.

For example, let $d_1, \cdots d_n$ $(n \gg 2)$ be disjoint plaintexts, k_{pub} be a public key, k_{sec} be the secret key corresponding to k_{pub}, e be an encrypted message, and $\sigma_{enc}(d_1, k_{pub}) = e$ hold. Suppose that an agent X knows one of the plaintexts $d_1, \cdots d_n$ is encrypted in the message e, and that X does not know the secret key k_{sec}. If X does not know at all which plaintext is encrypted in e, X thinks it possible that another plaintext $d_i\,(2 \leq i \leq n)$ is encrypted in e. In the model described above, this intuition is formalized as follows: $\sigma_{enc}(d_1, k_{pub}) = e$ holds at the actual world, $\sigma_{enc}(d_2, k_{pub}) = e$ holds at another world X consider possible, $\cdots$, and $\sigma_{enc}(d_n, k_{pub}) = e$ holds at another possible world. Note that the interpretation of the encryption function symbol σ_{enc} in this model varies from possible world to possible world. Since the interpretation of cryptographic function symbols, in general, cannot change as time passes, any cryptographic function symbol at a point has the same interpretation as those at any other point on the same run. Hence, it is natural to consider that the interpretation of cryptographic function symbols varies from run to run.

By employing an epistemic logic, we can express the *secrecy* of d_1 with respect to X as the following formula:

$$(\sigma_{enc}(d_1, k_{pub}) = e) \rightarrow \bigwedge_{1 \leq i \leq n} \mathbf{P}_X\ (\sigma_{enc}(d_i, k_{pub}) = e).$$

It should be noted that this formula is similar to the *total secrecy* defined in [3], where the secrecy of cryptography is not treated.

To deal formally with agents' knowledge of cryptographic functions in multi-agent systems, we consider a system $\mathcal{R}$ which satisfies the following conditions:

- the cryptographic functions denoted by the function symbols σ_{enc} and σ_{dec} are associated with each run of $\mathcal{R}$,
- for any interpretation ψ of σ_{enc} and σ_{dec}, there exists a run associated with ψ, and

- any constant symbol denotes the same fixed element of $\mathcal{D}$ in any possible world.

The second condition implies that any agent considers all the interpretations of σ_{enc} and σ_{dec} possible at time 0, because the *history* and the $func$ of any agent are empty at time 0.

This formalization allows us to consider the process whereby agents gradually acquire their partial knowledge of cryptographic functions by performing cryptographic actions. We assume that agents' knowledge of cryptographic functions is determined by their own cryptographic actions corresponding to the functions, that is, they never know the results of encryption/decryption until they actually perform the encryption/decryption actions. For example, if an agent X has never performed either $\langle encrypt, \langle d_1, k_{pub}\rangle, e, X\rangle$ or $\langle decrypt, \langle e, k_{sec}\rangle, d_1, X\rangle$ at a point (r, m), then X thinks it possible that the encrypted message e is obtained from the encryption of any plaintext d_i, because $r_X(m).func$ does not contain $\langle \epsilon, \langle d_1, k_{pub}\rangle, e\rangle$. Thus, before performing a cryptographic action, any agent considers that there are various possible interpretations of the cryptographic function symbols that do not agree with the results of the cryptographic actions the agent has not performed.

3 Protocol

3.1 Protocol Expressions and their Semantics

This section describes syntax and semantics for a protocol specification. We use sansserif letters for syntax, and *italic letters* for semantics. The syntax consists of data expressions, action expressions, and protocol expressions.

We define the following sets of symbols, which are mutually exclusive:

- a set I of *agent identifier symbols*,
- a set $\mathsf{K_{pub}}$ of *public key symbols*,
- a set $\mathsf{K_{sec}}$ of *secret key symbols*, and
- a set E of *encryption result symbols*.

Intuitively, they represent IDs of agents, public keys, secret keys, and the results of encryption/decryption, respectively. We assume that each element in $\mathsf{K_{pub}}$ is implicitly associated one-to-one with an element in $\mathsf{K_{sec}}$, that is, $\mathsf{k_{pub}}$, $\mathsf{k_{pub1}}$, $\mathsf{k_{pub2}}$, ..., $\mathsf{b_{pub1}}$, $\mathsf{b_{pub2}}$, ... in $\mathsf{K_{pub}}$ uniquely correspond to $\mathsf{k_{sec}}$, $\mathsf{k_{sec1}}$, $\mathsf{k_{sec2}}$, ..., $\mathsf{b_{sec1}}$, $\mathsf{b_{sec2}}$, ... in $\mathsf{K_{sec}}$, respectively. Suppose that $\mathsf{K} = \mathsf{K_{pub}} \cup \mathsf{K_{sec}}$, $\mathsf{C} = \mathsf{I} \cup \mathsf{K}$, and $\mathsf{D} = \mathsf{C} \cup \mathsf{E}$. Each $\mathsf{d} \in \mathsf{D}$ is referred to as a *data expression*. The identity relation on D is denoted by $\equiv$.

To denote action types, we introduce *action symbols*: $\mathsf{send_{norm}}$, $\mathsf{send_{ano}}$, $\mathsf{encrypt}$, and $\mathsf{decrypt}$. We also use a special action symbol $\mathsf{possess}$ to specify the agents' initial knowledge of data. Each *action expression* has one of the following forms.

$\mathsf{possess}(\mathsf{X}, \mathsf{c}_1, \ldots, \mathsf{c}_n)$	$(\mathsf{X} \in \mathsf{I}, \mathsf{c}_1, \ldots, \mathsf{c}_n \in \mathsf{C})$
$\mathsf{send_{norm}}(\mathsf{d}, \mathsf{X}, \mathsf{Y})$	$(\mathsf{d} \in \mathsf{D}, \mathsf{X}, \mathsf{Y} \in \mathsf{I})$
$\mathsf{send_{ano}}(\mathsf{d}, \mathsf{X}, \mathsf{Y})$	$(\mathsf{d} \in \mathsf{D}, \mathsf{X}, \mathsf{Y} \in \mathsf{I})$
$\mathsf{encrypt}(\langle \mathsf{d}, \mathsf{k}\rangle, \mathsf{e}, \mathsf{X})$	$(\mathsf{d} \in \mathsf{D}, \mathsf{k} \in \mathsf{K}, \mathsf{e} \in \mathsf{E}, \mathsf{X} \in \mathsf{I})$
$\mathsf{decrypt}(\langle \mathsf{e}, \mathsf{k}\rangle, \mathsf{d}, \mathsf{X})$	$(\mathsf{e} \in \mathsf{E}, \mathsf{k} \in \mathsf{K}, \mathsf{d} \in \mathsf{D}, \mathsf{X} \in \mathsf{I})$

The set of all the action expressions is denoted by ACT.

A *timed action expression* is a pair $\mathsf{A}@m$ consisting of an action expression A and time m with a *separating symbol* @. Any timed action expression with $\mathsf{possess}$ is restricted to the case $m = 0$. A *protocol expression* P is a finite sequence of timed action expressions $\mathsf{A}_0@m_0$, $\mathsf{A}_1@m_1$, ..., $\mathsf{A}_l@m_l$ satisfying the following conditions:

1. $\mathsf{A}_i \in \mathsf{ACT}$ for $i = 0, \ldots, l$, and $m_i \leq m_j$ if $i \leq j$.
2. For any $i > 0$, arguments of A_i listed below also appear in $\mathsf{A}_j@m_j$ for some $m_j < m_i$:
   - all its arguments if A_i is a communication action expression.
   - d_i, k_i, and X_i if A_i is of the form $\mathsf{encrypt}(\langle \mathsf{d}_i, \mathsf{k}_i\rangle, \mathsf{e}_i, \mathsf{X}_i)$.
   - e_i, k_i, and X_i if A_i is of the form $\mathsf{decrypt}(\langle \mathsf{e}_i, \mathsf{k}_i\rangle, \mathsf{d}_i, \mathsf{X}_i)$.

Note that the latter three conditions mean that agents must possess data before performing actions using them. We identify the order of the timed action expressions where the same time expression occurs. The *composition* of two protocol expressions P_1 and P_2, denoted by $\mathsf{P}_1 * \mathsf{P}_2$, is the protocol expression obtained by listing all the timed action expressions in P_1 and P_2 in chronological order.

Next, we define the semantics of a protocol expression. Recall that σ_{enc} and σ_{dec} are two binary function symbols denoting the encryption/decryption functions associated with each run. The conditions for the cryptographic functions described in Section 2.2 are represented by the following set Γ of equations.

$$\Gamma = \{ \quad \sigma_{\mathsf{dec}}(\sigma_{\mathsf{enc}}(\mathsf{x}, \mathsf{k_{pub}}), \mathsf{k_{sec}}) = \mathsf{x},$$
$$\sigma_{\mathsf{enc}}(\sigma_{\mathsf{dec}}(\mathsf{x}, \mathsf{k_{sec}}), \mathsf{k_{pub}}) = \mathsf{x},$$
$$\sigma_{\mathsf{dec}}(\sigma_{\mathsf{enc}}(\mathsf{x}, \mathsf{k_{sec}}), \mathsf{k_{pub}}) = \mathsf{x},$$
$$\sigma_{\mathsf{enc}}(\sigma_{\mathsf{dec}}(\mathsf{x}, \mathsf{k_{pub}}), \mathsf{k_{sec}}) = \mathsf{x}$$
$$\mid \mathsf{k_{pub}} \in \mathsf{K_{pub}}, \ \mathsf{k_{sec}} \in \mathsf{K_{sec}} \}$$

A *data assignment* ψ associated with each run r is a mapping that assigns an element in the data domain $\mathcal{D}$ to every $\mathsf{d} \in \mathsf{D}$, and the encryption/decryption functions associated with r to σ_{enc} and to σ_{dec}. For any term t constructed from $\mathsf{D} \cup \{\sigma_{\mathsf{enc}}, \sigma_{\mathsf{dec}}\}$, the assignment of an element in $\mathcal{D}$ to t by ψ is defined as usual.

We say that a run r with its data assignment ψ *follows* a protocol expression P at time m, denoted by $(r, m) \models \mathit{follow}(\mathsf{P})$, if the following conditions are satisfied:

- If $\mathsf{possess}(\mathsf{X}, \mathsf{c}_1, \ldots, \mathsf{c}_n)@0$ appears in P, then $r_X(0).\mathit{data} = \{\psi(\mathsf{c}_1), \ldots, \psi(\mathsf{c}_n)\}$.
- If $\mathsf{send}_{\mathsf{norm}}(\mathsf{d}, \mathsf{X}, \mathsf{Y})@i$ appears in P, then the action $\langle \mathit{send}_{norm}, \psi(\mathsf{d}), \psi(\mathsf{X}), \psi(\mathsf{Y})\rangle$ is performed between $(r, m+i-1)$ and $(r, m+i)$.
- If $\mathsf{send}_{\mathsf{ano}}(\mathsf{d}, \mathsf{X}, \mathsf{Y})@i$ appears in P, then the action $\langle \mathit{send}_{ano}, \psi(\mathsf{d}), \psi(\mathsf{X}), \psi(\mathsf{Y})\rangle$ is performed between $(r, m+i-1)$ and $(r, m+i)$.
- If $\mathsf{encrypt}(\langle \mathsf{d}, \mathsf{k}\rangle, \mathsf{e}, \mathsf{X})@i$ appears in P, then the action $\langle \mathit{encrypt}, \langle \psi(\mathsf{d}), \psi(\mathsf{k})\rangle, \psi(\mathsf{e}), \psi(\mathsf{X})\rangle$ is performed between $(r, m+i-1)$ and $(r, m+i)$.
- If $\mathsf{decrypt}(\langle \mathsf{e}, \mathsf{k}\rangle, \mathsf{d}, \mathsf{X})@i$ appears in P, then the action $\langle \mathit{decrypt}, \langle \psi(\mathsf{e}), \psi(\mathsf{k})\rangle, \psi(\mathsf{d}), \psi(\mathsf{X})\rangle$ is performed between $(r, m+i-1)$ and $(r, m+i)$.
- If a timed action expression $\mathsf{A}@i$ of a normal agent does not appear in P, then the normal agent does not perform the action between $(r, m+i-1)$ and $(r, m+i)$.
- If either d or k is not occurred in $\mathsf{A}_0@m_0, \cdots, \mathsf{A}_i@m_i$ of P, then no intruder agent can perform the cryptographic actions using both $\psi(\mathsf{d})$ and $\psi(\mathsf{k})$ between $(r, m+m_i)$ and $(r, m+m_i+1)$.

In such a case, we also say that P is *valid* with respect to ψ.

3.2 Standard Data Assignment

To argue secrecy syntactically, we need to avoid an interpretation of terms where two distinct expressions accidentally denote the same value. Thus, in this section, we describe a standard way of defining the data domain $\mathcal{D}$ and data assignments.

Let $\mathsf{Term}(\mathsf{C})$ be the set of all terms constructed from $\mathsf{C} \cup \{\sigma_{\mathsf{enc}}, \sigma_{\mathsf{dec}}\}$. We write the quotient set of $\mathsf{Term}(\mathsf{C})$ by the equivalence derived from Γ as $\mathsf{Term}(\mathsf{C})/\Gamma$, and the element in $\mathsf{Term}(\mathsf{C})/\Gamma$ containing a term t as $[\mathsf{t}]_\Gamma$. Let Θ be the set of all bijections from $\mathsf{Term}(\mathsf{C})/\Gamma$ to D satisfying $\theta([\mathsf{c}]_\Gamma) = \mathsf{c}$ for any $\theta \in \Theta$ and any $\mathsf{c} \in \mathsf{C}$. Given $\theta \in \Theta$, the *standard* data assignment ψ^θ is defined as the following:

- $\mathcal{D} = \mathsf{D}$.
- For any $\mathsf{d}_1, \mathsf{d}_2 \in \mathcal{D}$, $\psi^\theta(\sigma_{\mathsf{enc}})(\mathsf{d}_1, \mathsf{d}_2) = \theta([\sigma_{\mathsf{enc}}(\mathsf{t}_1, \mathsf{t}_2)]_\Gamma)$, where $\mathsf{t}_i \in \theta^{-1}(\mathsf{d}_i)$ $(i = 1, 2)$.
- For any $\mathsf{d}_1, \mathsf{d}_2 \in \mathcal{D}$, $\psi^\theta(\sigma_{\mathsf{dec}})(\mathsf{d}_1, \mathsf{d}_2) = \theta([\sigma_{\mathsf{dec}}(\mathsf{t}_1, \mathsf{t}_2)]_\Gamma)$, where $\mathsf{t}_i \in \theta^{-1}(\mathsf{d}_i)$ $(i = 1, 2)$.

Note that, in the above definition, $[\sigma_{\mathsf{enc}}(\mathsf{t}_1, \mathsf{t}_2)]_\Gamma$ and $[\sigma_{\mathsf{dec}}(\mathsf{t}_1, \mathsf{t}_2)]_\Gamma$ are uniquely determined whichever $\mathsf{t}_i \in \theta^{-1}(\mathsf{d}_i)$ is chosen. By the above construction, for any standard data assignment ψ, it holds that

- $\psi(\sigma_{\mathsf{enc}})$ and $\psi(\sigma_{\mathsf{dec}})$ are collision-free,
- all the equations in Γ hold for ψ,
- for any $\mathsf{d}_1, \mathsf{d}_2 \in \mathcal{D}$, $\mathsf{d}_1 \not\equiv \mathsf{d}_2$ implies $\psi(\mathsf{d}_1) \neq \psi(\mathsf{d}_2)$, and
- for any $\mathsf{t}_1, \mathsf{t}_2 \in \mathsf{Term}(\mathsf{C})$, $\Gamma \not\vdash \mathsf{t}_1 = \mathsf{t}_2$ implies $\psi(\mathsf{t}_1) \neq \psi(\mathsf{t}_2)$.

Moreover, we can prove the following basic property concerning secrecy.

Lemma 3.1 *For any standard data assignment ψ, intruder agents cannot know $\psi(\mathsf{c})$ by observing the execution of P if $\mathsf{c} \in \mathsf{C}$ does not appear in P syntactically.*

Proof It is easy to see that c occurs syntactically in t for any $\mathsf{t} \in [\mathsf{c}]_\Gamma$. Suppose c does not appear in P. Then every symbol in P denotes a value that is denoted by some c-free term in $\mathsf{Term}(\mathsf{C})$. Thus, all values obtained by the arbitrary repetition of encryption/decryption actions are also represented by c-free terms in $\mathsf{Term}(\mathsf{C})$, which cannot denote $\psi(\mathsf{c})$. ∎

4 Anonymity

In this section, we describe a method of verifying the anonymity of security protocols in the aforementioned model by using standard data assignments.

4.1 Detailed Example: Electronic Voting Protocols

To simplify the discussion, hereafter we consider the following simple protocol, which covers certain essential aspects of complex electronic voting protocols, and assume that there is only one intruder

agent I.

$\mathsf{vote1}(\mathsf{b_{sec1}}, \mathsf{v_1}, \mathsf{e'_1})$:
$\mathsf{encrypt}(\langle \mathsf{b_{sec1}}, \mathsf{k_{pub1}} \rangle, \mathsf{e_1}, \mathsf{A})$ @1
$\mathsf{send_{norm}}(\mathsf{e_1}, \mathsf{A}, \mathsf{V_1})$ @2
$\mathsf{decrypt}(\langle \mathsf{e_1}, \mathsf{k_{sec1}} \rangle, \mathsf{b_{sec1}}, \mathsf{V_1})$ @3
$\mathsf{encrypt}(\langle \mathsf{v_1}, \mathsf{b_{sec1}} \rangle, \mathsf{e'_1}, \mathsf{V_1})$ @4
$\mathsf{send_{ano}}(\mathsf{e'_1}, \mathsf{V_1}, \mathsf{C})$ @5

This voting protocol is constituted by the actions of normal agents A (the voting administrator), V_1, V_2, V_3 (voters), C (the vote collector), and the intruder agent I. Data expressing the names of candidates are denoted by agent identifier symbols $B_1, \ldots, B_5$. We use the meta variables $\mathsf{v_1}$, $\mathsf{v_2}$, $\mathsf{v_3}$, each denoting one of the candidates. The protocol is divided into two steps. In the first step, the administrator A encrypts a secret key $\mathsf{b_{sec1}}$ to obtain a message $\mathsf{e_1}$ by using the encryption function σ_{enc} and a public key $\mathsf{k_{pub1}}$. At the next time, A sends $\mathsf{e_1}$ to the voter V_1 through a normal communication channel. In the second step, V_1 decrypts the message $\mathsf{e_1}$ to obtain the key $\mathsf{b_{sec1}}$ by using the decryption function σ_{dec} and a secret key $\mathsf{k_{sec1}}$. Then V_1 encrypts the contents of vote $\mathsf{v_1}$ to obtain an encrypted message $\mathsf{e'_1}$ by using the encryption function σ_{enc} and the key $\mathsf{b_{sec1}}$ received from A. At the end of this protocol, V_1 sends $\mathsf{e'_1}$ to the collector C through an anonymous communication channel. In the same way, we define two voting protocols $\mathsf{vote2}(\mathsf{b_{sec2}}, \mathsf{v_2}, \mathsf{e'_2})$ and $\mathsf{vote3}(\mathsf{b_{sec3}}, \mathsf{v_3}, \mathsf{e'_3})$. Note that these protocols have parameters. In a later section, we consider *variant protocols* of $\mathsf{vote1}(\mathsf{b_{sec1}}, \mathsf{v_1}, \mathsf{e'_1})$ such as $\mathsf{vote1}(\mathsf{b_{sec2}}, \mathsf{v_2}, \mathsf{e'_2})$.

From the notation of these voting protocols, we omit the initial knowledge of data for brevity. We assume that each agent in these protocols possesses the following data before executing the protocols.

Agent	Initial Knowledge of Data
A	$\mathsf{A}, \mathsf{V_1}, \mathsf{V_2}, \mathsf{V_3}, \mathsf{C}, \mathsf{B_1}, \ldots, \mathsf{B_5}$, $\mathsf{k_{pub1}}, \mathsf{k_{pub2}}, \mathsf{k_{pub3}}, \mathsf{b_{pub1}}, \mathsf{b_{pub2}}, \mathsf{b_{pub3}}$, $\mathsf{b_{sec1}}, \mathsf{b_{sec2}}, \mathsf{b_{sec3}}$
V_1	$\mathsf{A}, \mathsf{V_1}, \mathsf{V_2}, \mathsf{V_3}, \mathsf{C}, \mathsf{B_1}, \ldots, \mathsf{B_5}$, $\mathsf{k_{pub1}}, \mathsf{k_{pub2}}, \mathsf{k_{pub3}}, \mathsf{b_{pub1}}, \mathsf{b_{pub2}}, \mathsf{b_{pub3}}$, $\mathsf{k_{sec1}}$
V_2	$\mathsf{A}, \mathsf{V_1}, \mathsf{V_2}, \mathsf{V_3}, \mathsf{C}, \mathsf{B_1}, \ldots, \mathsf{B_5}$, $\mathsf{k_{pub1}}, \mathsf{k_{pub2}}, \mathsf{k_{pub3}}, \mathsf{b_{pub1}}, \mathsf{b_{pub2}}, \mathsf{b_{pub3}}$, $\mathsf{k_{sec2}}$
V_3	$\mathsf{A}, \mathsf{V_1}, \mathsf{V_2}, \mathsf{V_3}, \mathsf{C}, \mathsf{B_1}, \ldots, \mathsf{B_5}$, $\mathsf{k_{pub1}}, \mathsf{k_{pub2}}, \mathsf{k_{pub3}}, \mathsf{b_{pub1}}, \mathsf{b_{pub2}}, \mathsf{b_{pub3}}$, $\mathsf{k_{sec3}}$
C	$\mathsf{A}, \mathsf{V_1}, \mathsf{V_2}, \mathsf{V_3}, \mathsf{C}, \mathsf{B_1}, \ldots, \mathsf{B_5}$, $\mathsf{k_{pub1}}, \mathsf{k_{pub2}}, \mathsf{k_{pub3}}, \mathsf{b_{pub1}}, \mathsf{b_{pub2}}, \mathsf{b_{pub3}}$
I	$\mathsf{A}, \mathsf{V_1}, \mathsf{V_2}, \mathsf{V_3}, \mathsf{C}, \mathsf{B_1}, \ldots, \mathsf{B_5}$, $\mathsf{k_{pub1}}, \mathsf{k_{pub2}}, \mathsf{k_{pub3}}, \mathsf{b_{pub1}}, \mathsf{b_{pub2}}, \mathsf{b_{pub3}}$

4.2 Anonymity Verification Method

This section defines the anonymity of protocols in this paper. Then we formally introduce notions for anonymity verification, namely, secrecy and key-independence. We also introduce a proof method based on a swap of secret keys.

Definition 4.1 The two sequences $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ of the parameters occurring in a protocol $\mathsf{P}(\boldsymbol{x_1}, \boldsymbol{x_2})$ are called *interchangeable* with respect to the intruder agent I at time m, if the formula
$\bigcirc^{-m} follow(\mathsf{P}(\boldsymbol{x_1}, \boldsymbol{x_2})) \rightarrow \mathbf{P}_I \bigcirc^{-m} follow(\mathsf{P}(\boldsymbol{x_2}, \boldsymbol{x_1}))$
is satisfied at point (r, m) for any r.

For example, suppose $\boldsymbol{x_1} = (\mathsf{b_{sec1}}, \mathsf{v_1}, \mathsf{e'_1})$ and $\boldsymbol{x_2} = (\mathsf{b_{sec2}}, \mathsf{v_2}, \mathsf{e'_2})$. Let $\mathsf{P}(\boldsymbol{x_1}, \boldsymbol{x_2})$ be the composed protocol $\mathsf{vote1}(\mathsf{b_{sec1}}, \mathsf{v_1}, \mathsf{e'_1}) * \mathsf{vote2}(\mathsf{b_{sec2}}, \mathsf{v_2}, \mathsf{e'_2})$, and $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ in $\mathsf{P}(\boldsymbol{x_1}, \boldsymbol{x_2})$ be interchangeable with respect to I at time m. If the formula $\bigcirc^{-m} follow(\mathsf{vote1}(\mathsf{b_{sec1}}, \mathsf{v_1}, \mathsf{e'_1}) * \mathsf{vote2}(\mathsf{b_{sec2}}, \mathsf{v_2}, \mathsf{e'_2}))$ is satisfied at (r, m), $\bigcirc^{-m} follow(\mathsf{vote1}(\mathsf{b_{sec2}}, \mathsf{v_2}, \mathsf{e'_2}) * \mathsf{vote2}(\mathsf{b_{sec1}}, \mathsf{v_1}, \mathsf{e'_1}))$ is satisfied at a point I considers possible at (r, m). Thus, I does not know which of the voters V_1 and V_2 sent the vote $\mathsf{v_1}$. This is the anonymity we want to show in the next section.

Next, we introduce notions for verifying anonymity. We call data *secret* if the intruder agent I does not know them. To represent the secrecy of a secret key $\mathsf{k_{sec}}$ with respect to I, we use the formula $secret(\mathsf{k_{sec}})$. Formally, $secret(\mathsf{k_{sec}})$ is true at a point (r, m), if the value of $\mathsf{k_{sec}}$ is not in $r_I(m).data$. Hereafter we assume that the interpretation of terms follows a standard data assignment. Also, the satisfaction of formulas is considered in an interpreted system with the truth assignment of *secret* and *follow* defined so far, and with all runs obtained by all protocol expressions.

Then we present a basic way of proving secrecy.

Definition 4.2 Two protocol expressions are *key-independent* if any secret key symbol occurring in one protocol expression does not occur in the other.

An important property of key-independence is that the composition of key-independent protocol expressions also preserves the secrecy of secret keys.

Proposition 4.3 *Let* P *and* Q *be key-independent protocol expressions. Suppose that a key* k *appears syntactically in* P*, and that the formula* $follow(\mathsf{P}) \rightarrow \Box(secret(\mathsf{k}))$ *holds. Then* $follow(\mathsf{P} * \mathsf{Q}) \rightarrow \Box(secret(\mathsf{k}))$ *also holds.*

Proof This property is proved by a similar syntactic argument to that of Lemma 3.1. ∎

Finally, we introduce a simulation proof method for anonymity based on a swap of secret key symbols.

Definition 4.4 A run r' *simulates* r if $r'_I(m) = r_I(m)$ for any m.

Definition 4.5 Let $\mathsf{b_{sec1}}$, $\mathsf{b_{sec2}}$, $\mathsf{k_{sec1}}$, and $\mathsf{k_{sec2}}$ be key symbols, and ψ^θ be a standard data assignment. The *key swap* of ψ^θ *between* $\mathsf{b_{sec1}}$ *and* $\mathsf{b_{sec2}}$ *guarded by* $\mathsf{k_{sec1}}$ and $\mathsf{k_{sec2}}$ is a standard data assignment $\psi^{\theta'}$ defined by the following θ':

$$\theta'([\mathsf{t}]_\Gamma) = \theta([\mathsf{swap(t)}]_\Gamma) \quad (\text{for } i = 1, 2),$$

where $\mathsf{swap(t)}$ is the term obtained from the minimum[1] term $\hat{\mathsf{t}}$ in $[\mathsf{t}]_\Gamma$ by replacing every occurrence of $\sigma_{\mathsf{enc}}(\mathsf{b_{sec}}i, \mathsf{k_{pub}}j)$ in $\hat{\mathsf{t}}$ with $\sigma_{\mathsf{enc}}(\mathsf{b_{sec}}3-i, \mathsf{k_{pub}}j)$ for $i, j = 1, 2$.

Proposition 4.6 *Suppose that a run r with a standard data assignment ψ^θ follows a protocol expression* P *at time* 0. *Assume that key symbols* $\mathsf{b_{sec1}}$, $\mathsf{b_{sec2}}$, $\mathsf{k_{sec1}}$, *and* $\mathsf{k_{sec2}}$ *are secret in r. Let* $\mathsf{P'}$ *be a protocol expression which is identical to* P *with respect to communication actions. Assume also that* $\mathsf{P'}$ *is valid with respect to the key swap* $\psi^{\theta'}$ *of* ψ^θ *between* $\mathsf{b_{sec1}}$ *and* $\mathsf{b_{sec2}}$ *guarded by* $\mathsf{k_{sec1}}$ *and* $\mathsf{k_{sec2}}$. *Then, there exists a run r' with* $\psi^{\theta'}$ *such that r' simulates r and* $(r', 0) \models \mathit{follow}(\mathsf{P'})$ *holds.*

Proof We assume there does not exist such a simulation, and derive a contradiction. Since communication actions are the same, the first action that cannot be simulated must be an encryption/decryption action performed by I. First, suppose that I performs

- an action $\mathsf{encrypt}(\langle \mathsf{d}, \mathsf{k}\rangle, \mathsf{e}, \mathsf{l})$ in r, and
- an action $\mathsf{encrypt}(\langle \mathsf{d}, \mathsf{k}\rangle, \mathsf{e'}, \mathsf{l})$ in r'.

From the definition of θ', d must be $\mathsf{b_{sec1}}$ or $\mathsf{b_{sec2}}$ if $\mathsf{e} \neq \mathsf{e'}$. However, I cannot perform that action in r, because $\mathsf{b_{sec1}}$ and $\mathsf{b_{sec2}}$ are secret. Next, suppose that I performs

- an action $\mathsf{decrypt}(\langle \mathsf{e}, \mathsf{k}\rangle, \mathsf{d}, \mathsf{l})$ in r, and
- an action $\mathsf{decrypt}(\langle \mathsf{e}, \mathsf{k}\rangle, \mathsf{d'}, \mathsf{l})$ in r'.

Similarly, from the definition of θ', k must be $\mathsf{k_{sec1}}$ or $\mathsf{k_{sec2}}$ if $\mathsf{d} \neq \mathsf{d'}$. However, I cannot perform that action in r, because $\mathsf{k_{sec1}}$ and $\mathsf{k_{sec2}}$ are secret. This leads to a contradiction, and thus the result follows. ∎

[1] Roughly speaking, the minimum term is the normal form of t when we use equations in Γ as a left-to-right rewrite rules.

4.3 Deriving Anonymity by Interchangeability

In this section, we discuss a method of deriving anonymity by using the partial knowledge of cryptographic functions in the example of the voting protocols described above: $\mathsf{vote1}(\mathsf{b_{sec1}}, \mathsf{v}_1, \mathsf{e}'_1)$ and $\mathsf{vote2}(\mathsf{b_{sec2}}, \mathsf{v}_2, \mathsf{e}'_2)$. Note that these two protocol expressions are key-independent. Let P be the protocol expression $\mathsf{vote1}(\mathsf{b_{sec1}}, \mathsf{v}_1, \mathsf{e}'_1) * \mathsf{vote2}(\mathsf{b_{sec2}}, \mathsf{v}_2, \mathsf{e}'_2)$, and Q be a protocol expression that is key-independent of P. Suppose that the execution of these protocols terminates at time $m_{\max}$. We assume the secrecy of secret keys:

- $\mathit{follow}(\mathsf{vote1}(\mathsf{b_{sec1}}, \mathsf{v}_1, \mathsf{e}'_1)) \rightarrow \Box(\mathit{secret}(\mathsf{b_{sec1}}) \wedge \mathit{secret}(\mathsf{k_{sec1}}))$, and
- $\mathit{follow}(\mathsf{vote2}(\mathsf{b_{sec2}}, \mathsf{v}_2, \mathsf{e}'_2)) \rightarrow \Box(\mathit{secret}(\mathsf{b_{sec2}}) \wedge \mathit{secret}(\mathsf{k_{sec2}}))$.

By Proposition 4.3, we obtain

- $\mathit{follow}(\mathsf{P} * \mathsf{Q}) \rightarrow \Box(\mathit{secret}(\mathsf{b_{sec1}}) \wedge \mathit{secret}(\mathsf{k_{sec1}}))$, and
- $\mathit{follow}(\mathsf{P} * \mathsf{Q}) \rightarrow \Box(\mathit{secret}(\mathsf{b_{sec2}}) \wedge \mathit{secret}(\mathsf{k_{sec2}}))$.

Let r be such a run that $(r, 0) \models \mathit{follow}(\mathsf{P} * \mathsf{Q})$. Then we have $(r, m_{\max}) \models \mathit{secret}(\mathsf{b_{sec1}}) \wedge \mathit{secret}(\mathsf{k_{sec1}}) \wedge \mathit{secret}(\mathsf{b_{sec2}}) \wedge \mathit{secret}(\mathsf{k_{sec2}})$. Since agents cannot encrypt or decrypt what they do not possess in this model, the following actions are not performed by I.

- $\mathsf{encrypt}(\langle \mathsf{b_{sec1}}, \mathsf{k_{pub1}}\rangle, ...)$
- $\mathsf{decrypt}(\langle \mathsf{e}_1, \mathsf{k_{sec1}}\rangle, ...)$
- $\mathsf{encrypt}(\langle \mathsf{b_{sec2}}, \mathsf{k_{pub2}}\rangle, ...)$
- $\mathsf{decrypt}(\langle \mathsf{e}_2, \mathsf{k_{sec2}}\rangle, ...)$

Therefore, I does not acquire the following knowledge of cryptographic functions in $r_I(m_{\max}).\mathit{func}$.

- $\langle \epsilon, \langle b_{sec1}, k_{pub1}\rangle, e_1\rangle$
- $\langle \delta, \langle e_1, k_{sec1}\rangle, b_{sec1}\rangle$
- $\langle \epsilon, \langle b_{sec2}, k_{pub2}\rangle, e_2\rangle$
- $\langle \delta, \langle e_2, k_{sec2}\rangle, b_{sec2}\rangle$

Thus, I thinks there are various possible interpretations of the function symbols σ_{enc} and σ_{dec} which do not agree with this function knowledge. In fact, let us consider the protocol expression $\mathsf{vote1}(\mathsf{b_{sec2}}, \mathsf{v}_2, \mathsf{e}'_2) * \mathsf{vote2}(\mathsf{b_{sec1}}, \mathsf{v}_1, \mathsf{e}'_1)$, denoted by $\mathsf{P'}$. Then it follows from Proposition 4.6 that we

can construct a run r' that simulates r in I's viewpoint, and thus, we have $(r', 0) \models \mathit{follow}(\mathsf{P}' * \mathsf{Q})$. Since $r'_I(m_{\max}) \sim_I r_I(m_{\max})$ holds, we have $(r, m_{\max}) \models \mathbf{P}_I \bigcirc^{-m_{\max}} \mathit{follow}(\mathsf{P}' * \mathsf{Q})$. Hence, we obtain

$$\begin{aligned}(\bigcirc^{-m_{\max}} \mathit{follow}(&\mathsf{vote1}(\mathsf{b}_{\mathsf{sec1}}, \mathsf{v}_1, \mathsf{e}'_1) \\ &* \mathsf{vote2}(\mathsf{b}_{\mathsf{sec2}}, \mathsf{v}_2, \mathsf{e}'_2) * \mathsf{Q})) \\ \rightarrow \mathbf{P}_I \bigcirc^{-m_{\max}} \mathit{follow}(&\mathsf{vote1}(\mathsf{b}_{\mathsf{sec2}}, \mathsf{v}_2, \mathsf{e}'_2) \\ &* \mathsf{vote2}(\mathsf{b}_{\mathsf{sec1}}, \mathsf{v}_1, \mathsf{e}'_1) * \mathsf{Q}),\end{aligned}$$

that is, $(\mathsf{b}_{\mathsf{sec1}}, \mathsf{v}_1, \mathsf{e}'_1)$ and $(\mathsf{b}_{\mathsf{sec2}}, \mathsf{v}_2, \mathsf{e}'_2)$ occurring in $\mathsf{P} * \mathsf{Q}$ are interchangeable with respect to I. Let Q be the protocol expression $\mathsf{vote3}(\mathsf{b}_{\mathsf{sec3}}, \mathsf{v}_3, \mathsf{e}'_3)$. As Q is key-independent of P, we obtain

$$\begin{aligned}(\bigcirc^{-m_{\max}} \mathit{follow}(&\mathsf{vote1}(\mathsf{b}_{\mathsf{sec1}}, \mathsf{v}_1, \mathsf{e}'_1) \\ &* \mathsf{vote2}(\mathsf{b}_{\mathsf{sec2}}, \mathsf{v}_2, \mathsf{e}'_2) \\ &* \mathsf{vote3}(\mathsf{b}_{\mathsf{sec3}}, \mathsf{v}_3, \mathsf{e}'_3)) \\ \rightarrow \mathbf{P}_I \bigcirc^{-m_{\max}} \mathit{follow}(&\mathsf{vote1}(\mathsf{b}_{\mathsf{sec2}}, \mathsf{v}_2, \mathsf{e}'_2) \\ &* \mathsf{vote2}(\mathsf{b}_{\mathsf{sec1}}, \mathsf{v}_1, \mathsf{e}'_1) \\ &* \mathsf{vote3}(\mathsf{b}_{\mathsf{sec3}}, \mathsf{v}_3, \mathsf{e}'_3)).\end{aligned} \tag{1}$$

In the same way, we have

$$\begin{aligned}(\bigcirc^{-m_{\max}} \mathit{follow}(&\mathsf{vote1}(\mathsf{b}_{\mathsf{sec2}}, \mathsf{v}_2, \mathsf{e}'_2) \\ &* \mathsf{vote2}(\mathsf{b}_{\mathsf{sec1}}, \mathsf{v}_1, \mathsf{e}'_1) \\ &* \mathsf{vote3}(\mathsf{b}_{\mathsf{sec3}}, \mathsf{v}_3, \mathsf{e}'_3)) \\ \rightarrow \mathbf{P}_I \bigcirc^{-m_{\max}} \mathit{follow}(&\mathsf{vote1}(\mathsf{b}_{\mathsf{sec2}}, \mathsf{v}_2, \mathsf{e}'_2) \\ &* \mathsf{vote2}(\mathsf{b}_{\mathsf{sec3}}, \mathsf{v}_3, \mathsf{e}'_3) \\ &* \mathsf{vote3}(\mathsf{b}_{\mathsf{sec1}}, \mathsf{v}_1, \mathsf{e}'_1)).\end{aligned} \tag{2}$$

It follows from (1) and (2) that

$$\begin{aligned}(\bigcirc^{-m_{\max}} \mathit{follow}(&\mathsf{vote1}(\mathsf{b}_{\mathsf{sec1}}, \mathsf{v}_1, \mathsf{e}'_1) \\ &* \mathsf{vote2}(\mathsf{b}_{\mathsf{sec2}}, \mathsf{v}_2, \mathsf{e}'_2) \\ &* \mathsf{vote3}(\mathsf{b}_{\mathsf{sec3}}, \mathsf{v}_3, \mathsf{e}'_3)) \\ \rightarrow \mathbf{P}_I \bigcirc^{-m_{\max}} \mathit{follow}(&\mathsf{vote1}(\mathsf{b}_{\mathsf{sec2}}, \mathsf{v}_2, \mathsf{e}'_2) \\ &* \mathsf{vote2}(\mathsf{b}_{\mathsf{sec3}}, \mathsf{v}_3, \mathsf{e}'_3) \\ &* \mathsf{vote3}(\mathsf{b}_{\mathsf{sec1}}, \mathsf{v}_1, \mathsf{e}'_1)).\end{aligned}$$

In this way, we obtain the anonymity of the protocol.

5 Discussion

The computational aspects of knowledge are also important in terms of capturing a wide variety of problems in a framework of reasoning about knowledge. The limitation of agents' computational abilities has been examined in many studies. For example, in the algorithmic knowledge approach [7], agents can possess algorithms in their local states, and the limitation of agents' computational abilities is expressed by the outputs of the algorithms. However, it should be noted that this approach is not suitable for expressing the process whereby agents gradually acquire a partial knowledge of cryptographic functions.

On the other hand, our approach describes the limitation of agents' computational abilities as two conditions.

First, the cryptographic operations available to agents are restricted. We consider only the two cryptographic functions σ_{enc} and σ_{dec}, and any internal action is forbidden except for the cryptographic actions corresponding to these functions. This restriction means that attack is limited to decryption actions using secret keys or a huge number of repeated encryption actions.

Second, we restrict the cryptographic actions that agents can perform: agents cannot encrypt or decrypt a message if they do not possess the message or the key. As agents cannot encrypt or decrypt random data, this restriction may be strong. To model cryptographic actions using random data, we need a probabilistic approach, in which secrecy and anonymity are expressed by employing probability measures on the runs of systems. A way of introducing probability to a multi-agent system is presented in [6] and [5].

Furthermore, in the model where agents' knowledge of cryptographic functions is determined by their own cryptographic actions, the following condition on agents' computational abilities are implicitly assumed: agents' computation power is too weak to derive any knowledge of secret plaintexts or keys from encryption actions using other plaintexts and decryption actions using other secret keys.

6 Conclusion

We have introduced a method for formalizing the partial knowledge of cryptographic functions, and have shown that this framework enables us to verify the anonymity of security protocols. Although this paper has dealt only with a simple security protocol, it provides a starting point for discussing the knowledge of complex security protocols. Our future research will provide a computational justification for our verification method in a similar way as in [1], and employ probability to discuss the knowledge of cryptographic functions.

Acknowledgements

We thank the members of Computing Theory Research Group at NTT and Hagiya Laboratory for helpful suggestions.

References

[1] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, Vol. 15, No. 2, pp. 103 – 127.

[2] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptography–AUSCRYPT '92*, pp. 244–251, 1992. LNCS 718.

[3] J. Y. Halpern and K. R. O'Neill. Secrecy in multiagent systems. In *CSFW 2002*, pp. 32–48, 2002.

[4] J. Y. Halpern and K. R. O'Neill. Anonymity and information hiding in multiagent systems. In *CSFW 2003*, pp. 75–88, 2003.

[5] J. Y. Halpern and K. R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, Vol. 13, No. 3, pp. 483–514, 2005.

[6] J. Y. Halpern and M. R. Tuttle. Knowledge, probability, and adversaries. *Journal of the ACM*, Vol. 40, No. 4, pp. 917–962, 1993.

[7] Y. Moses J. Y. Halpern and M. Y. Vardi. Algorithmic knowledge. In *Proceedings of the 5th Conference on Theoretical Aspects of Reasoning about Knowledge*, pp. 255–266, 1994.

[8] Y.Moses R. Fagin, J. Y. Halpern and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[9] P. F. Syverson and S. G. Stubblebine. Group principals and the formalization of anonymity. In *Proceedings of the World Congress on Formal Methods in the Development of Computing Systems*, Vol. I, pp. 814–833, 1999.

[10] R. van der Meyden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *CSFW 2004*, pp. 280–291, 2004.