Policy-Based Secrecy in the Runs & Systems Framework and Controlled Query Evaluation[*]

Joachim Biskup and Cornelia Tadros

Technische Universität Dortmund, Germany
{biskup,tadros}@ls6.cs.uni-dortmund.de

Abstract. Controlled Query Evaluation (CQE) ensures the confidentiality of information in the context of information systems where a user can gain information beyond the mere representation of released data. To date, CQE provides a host of enforcement protocols under different system parameters for client-server architectures. In this paper, we embed the CQE approach in the Runs & Systems model and relate confidentiality preservation to the notions of secrecy proposed by Halpern and O'Neill in the context of multiagent systems (MAS) as a unified framework. In particular, we introduce the novel notion of policy-based secrecy, and we show how to state a common confidentiality requirement of CQE in their framework. By this, we lay a foundation to make use of the various inference control protocols of CQE in MAS scenarios and also to extend the CQE confidentiality model to MAS. Further by adopting Halpern and O'Neill's unified and general framework we ease the comparison to other approaches in the future.

1 Introduction

Controlled Query Evaluation (CQE) is a framework for inference control in information systems. Inference control in general aims not to leak sensitive information to a user who can reason about the information revealed to him. The CQE framework supports client-server architectures with a single data provider and several clients and offers controlled update transactions as well as queries [1, 2] in logic-oriented information systems. In this paper, we relate the CQE approach [2] and the formal model of Halpern and O'Neill [3] for secrecy in multiagent systems (MAS), which is general enough to be comparable to other approaches and expressive enough to give options for different agent reasoning or policy models.

Halpern and O'Neill proposed *Runs & Systems* (introduced by Fagin et al. in [4]) as a general model for formulating notions of secrecy in MAS [3]. A variety of existing notions have been reformulated in this framework by Halpern and O'Neill in the same paper, i.e., *separability* [5], *generalized noninterference* [6],

[*] Notice for the use of this material The copyright of this material is retained by the Information Processing Society of Japan (IPSJ). This material is published on this web site with the agreement of the author (s) and the IPSJ. Please be complied with Copyright Law of Japan and the Code of Ethics of the IPSJ if any users wish to reproduce, make derivative work, distribute or make available to the public any part or whole thereof. All Rights Reserved, Copyright (C) Information Processing Society of Japan

asynchronous separability [7] and *asynchronous generalized noninterference* [7]. Furthermore, Halpern and O'Neill described several notions of *anonymity* in this framework in another work [8], supporting the choice of their framework and its expressiveness. Also, the basic framework has been augmented with probability or plausibility measures and corresponding notions of secrecy.

In line with Halpern and O'Neill's work, in this paper we reformulate CQE [2] in their framework. Firstly, this reformulation is another step towards extending CQE to multiagent systems in future work (cf. [9] as a preliminary draft). The features of the Runs & Systems framework to incorporate other forms of knowledge representations, e.g., probability or plausibility measures, are essential for reasoning or information sharing etc. in multiagent systems. Moreover, the logic of knowledge and time over the Runs & Systems model offers an intuitive way to specify confidentiality properties. Secondly, we hope to give a basis for comparison of CQE with other approaches of confidentiality preservation.

As our main original contribution, we propose another notion of secrecy, called *policy-based secrecy*, which fulfills the following objectives:

1. *policy-based secrecy* can be applied to any system described in the Runs & Systems model (e.g., not only to systems in the CQE setting); thus, policy-based secrecy can be applied not only to client-server architectures as present in CQE but to general MAS architectures
2. *policy-based secrecy* separates the different aspects of an information hiding property: (a) "**What** information needs to be hidden?", (b) "**Who** does it need to be hidden from?", (c) "**How well** does it need to be hidden?" and (d) "**When** does it need to be hidden (e.g., on what conditions)?" ((a)–(c) are taken from [8]); thus, policy-based secrecy can be easily adapted to other security needs
3. *policy-based secrecy* bases upon Halpern and O'Neill's notions of secrecy from [3]; in particular, policy-based secrecy is compliant with their framework
4. the notion of confidentiality preservation in CQE [10] is an instance of policy-based secrecy; thus, policy-based secrecy might be enforced by – possibly enhanced – CQE protocols in MAS

As a further contribution, we give a syntactic representation of policy-based secrecy in the logic of knowledge and time over the Runs & Systems model like in [3].

This paper is organized as follows. In Section 2, we introduce policy-based secrecy and compare it to secrecy notions in Halpern and O'Neill's framework. In Section 3, we see that confidentiality preservation in the CQE context ([10]) is a special case of policy-based secrecy. In Section 4, policy-based secrecy is described in terms of the logic of knowledge and time presented by [3]. There are also other approaches which propose a unified framework to define secrecy properties: In Section 5, we focus on one approach – *function views* [11] which use a set of functions as a description of system behavior to give a formal framework for security verification of a system. By this, we illustrate how we can apply and compare other approaches of confidentiality preservation to CQE easily by policy-based secrecy. Finally, in our conclusion in Section 6, we hint how the model developed in this paper opens new ways to apply or extend CQE under different aspects.

2 Policy-Based Secrecy

In [3] the information accessible to an agent is encoded in his local state in the description of the system. A system is statically described by a global state $(s_e, s_1, \ldots, s_n)$, i.e., the collection of all agent's local states $s_1, \ldots, s_n$ and optionally an environment state s_e, and dynamically as a set of *runs*. A *run* r is a function from discrete time $\mathbb{N}_0$ to global states, i.e., one possible evolvement of the system over time. A system $\mathcal{R}$ is a collection of runs and the set $\mathcal{PT}(\mathcal{R}) := \{(r, m) \mid r \in \mathcal{R}, m \in \mathbb{N}_0\}$ is called the *points* of $\mathcal{R}$. The current view of agent i on the system $\mathcal{R}$ at time m in a run r is

$$\mathcal{K}_i\,(r, m) := \{(r', m') \in \mathcal{PT}(\mathcal{R}) \mid r_i(m) = r'_i(m')\},$$

where $r_i(m)$ denotes the local state of agent i in the global state $r(m)$. In words, $\mathcal{K}_i\,(r, m)$ is the set of points in which agent i has the same local state as in (r, m). The set $\mathcal{K}_i\,(r, m)$ is called *i-information set* and models the reasoning of agent i about the system at point (r, m). In run r at time m all possible system states from agent i's point of view are $\mathcal{K}_i(r, m)$. If the reasoning of an agent i is expressed by $\mathcal{K}_i$, one assumes implicitly that agent i knows the system specification $\mathcal{R}$ and can determine $\mathcal{K}_i$.

Halpern and O'Neill weaken their notion *total secrecy*, which permits no information-flow at all, in particular under two aspects, **what** to protect by *total f-secrecy* as well as *C-secrecy* and **when** to protect by *C-secrecy*. These notions are still considered separately from each other in their work. In this section we generalize these two notions and propose a generic notion "policy-based" secrecy which will turn out to be suitable to express some confidentiality requirements in the CQE context throughout Section 3. Afterwards, we compare policy-based secrecy to Halpern and O'Neill's aforementioned notions of secrecy.

In our CQE-related proposal, an agent j *locally declares* in each point $(r, m) \in \mathcal{PT}(\mathcal{R})$ which information about his state an agent i – the potential attacker – must consider possible in a *j-possibility policy*:

Definition 1 (j-Possibility Policy). *A j-possibility policy is a function policy : $\mathcal{PT}(\mathcal{R}) \to \mathcal{P}(\mathcal{P}(\mathcal{P}(\mathcal{PT}(\mathcal{R}))))$ such that policy$(r, m) := \{I_1, I_2 \ldots\}$ contains sets I_k of j-information sets, where $\mathcal{P}$ denotes the power set operation. The sets I_k and policy(r, m) may be infinite.*

The idea behind the formalisms of Definition 1 is: To talk about a state of agent j we are bound to talk about j-information sets. An I_k contains (packed in j-information sets S) the set of all points in which j's state has properties of interest, e.g., agent j stored a relevant data item. We will speak about an I_k as *security-relevant* information or shortly *relevant* information. In the context of confidentiality preservation, I_k is no confidential information by itself, but essential to protect other confidential information and in this sense relevant information. For instance, if the fact that a patient has cancer is confidential, then the fact that he has no cancer is relevant information. Concrete j-possibility policies are demonstrated in the following example from the database area:

Example 1. An agent 1 stored information in form of propositions a and b in an internal database. E.g., his state description (a) can be read as "a holds and b doesn't hold". So the set $States_1 := \{(), (a), (b), (a, b)\}$ denotes all his possible states. Now agent 1 can declare some possibility policies:

- $policy_1(r, m) := \{\{\mathcal{K}_1\,(r_a, m_a)\}, \{\mathcal{K}_1\,(r_b, m_b)\}\}$, where (r_a, m_a) and (r_b, m_b) are points at which agent 1 is in state (a) and in state (b), respectively. The states (a) and (b) are both declared relevant.
- $policy_2(r, m) := \{\{\mathcal{K}_1\,(r_a, m_a), \mathcal{K}_1\,(r_b, m_b)\}\}$. The relevant information is that agent 1 is in state (a) *or* in state (b).
- $policy_3(r, m) := \{\{\mathcal{K}_1\,(r_a, m_a), \mathcal{K}_1\,(r_b, m_b)\}, \{\mathcal{K}_1\,(r_{ab}, m_{ab})\}\}$, where (r_{ab}, m_{ab}) is a point, at which agent 1 is in state (a, b). Relevant information is that agent 1 is in state (a) *or* in state (b), like in $policy_2$; additionally, state (a, b) is relevant.

In our example, all policies do not depend on the current run r and time m and are thus rather a global than a local declaration. This is not necessary but simplifies our presentation. In the context of confidentiality preservation these possibility policies get a more intuitive meaning:

- $policy_1$ requires the attacker to consider state (a) and state (b) of agent 1 possible. Thus, if the fact $\neg a \vee b$ holds in agent 1's database (corresponding to the set of states $States_1 \setminus \{(a)\} = \{(), (b), (a, b)\}$), this situation must be kept confidential; similarly, if the fact $a \vee \neg b$ holds in the database (corresponding to the set of states $States_1 \setminus \{(b)\} = \{(), (a), (a, b)\}$), this situation must be kept confidential as well.
- $policy_2$ requires the attacker to consider state (a) or state (b) of agent 1 possible. Thus, if the fact $a \equiv b$ holds in the database (corresponding to the set of states $States_1 \setminus \{(a), (b)\} = \{(), (a, b)\}$), this situation must be kept confidential.
- $policy_3$ like in $policy_2$ aims to keep the situation, in which the fact $a \equiv b$ holds in agent 1's database, confidential. Additionally, the attacker is required to consider state (a, b) possible. Thus, if the fact $\neg a \vee \neg b$ holds in the database (corresponding to the set of states $States_1 \setminus \{(a, b)\} = \{(), (a), (b)\}$), this situation must be kept confidential as well.

In words, a j-possibility policy describes a set of pieces of information $\{I_1, \ldots, I_n\}$ agent i, the attacker, should not rule out about agent j's state. Thus in every system state $(r, m) \in \mathcal{PT}(\mathcal{R})$ and for every piece of information I_k of the local policy $policy(r, m)$, describing a set of agent j's local states by a corresponding set of points $\bigcup_{S \in I_k} S$, there is a possible system state from agent i's point of view

$(r', m') \in \mathcal{K}_i(r, m)$ in which agent j's local state carries information I_k (i.e., $(r', m') \in \bigcup_{S \in I_k} S$). This is enforced via policy-based secrecy:

Definition 2 (Policy-based Secrecy). *If policy is a j-possibility policy, agent j maintains policy-based secrecy with respect to agent i in $\mathcal{R}$ if, for all points $(r, m) \in \mathcal{PT}(\mathcal{R})$ and for all $I_k \in policy(r, m)$:*

$$\mathcal{K}_i\,(r, m) \cap \bigcup_{S \in I_k} S \neq \emptyset$$

In the following we write *policy*-based secrecy to point out the used j-possibility policy *policy*. We continue Example 1 to illustrate policy-based secrecy:

Example 2. Another agent 2 – the potential attacker – can pose queries to agent 1, written as propositional formulas over the alphabet $\mathcal{A} = \{a, b\}$. The internal database serves as a propositional interpretation to evaluate the queries. Consider a system consisting of the following runs, in which agent 2 queries for a at time $m = 0$ and receives the answer in return at time $m = 1$. We write global states as (s_1, s_2) and agent 2's local states as $(-)$ and $(+)$, which mean agent 2 received a negative and positive answer, respectively.

$$\begin{array}{rllll}
 & m = 0 & m = 1 & \ldots & \\
\mathcal{R} := \{r_1 = \langle & ((), ()), & ((), (-)), & \ldots & \rangle, \\
r_2 = \langle & ((a), ()), & ((a), (+)), & \ldots & \rangle, \\
r_3 = \langle & ((b), ()), & ((b), (-)), & \ldots & \rangle, \\
r_4 = \langle & ((a, b), ()), & ((a, b), (+)), & \ldots & \rangle\}
\end{array}$$

Now, we discuss the preservation of the policies defined in Example 1 in the sense of Definition 2:

- $policy_1$ is violated in $(r_1, 1)$ because $\mathcal{K}_2\,(r_1, 1) \cap \mathcal{K}_1\,(r_a, m_a) = \{(r_1, 1), (r_3, 1), (r_1, 2), (r_3, 2), \ldots\} \cap \{(r_2, 0), (r_2, 1), \ldots\} = \emptyset$ with $policy_1(r, m) := \{\{\mathcal{K}_1\,(r_a, m_a)\}, \{\mathcal{K}_1\,(r_b, m_b)\}\}$, which says that in state $(-)$ agent 2 can rule out state (a) of agent 1
- in contrast, $policy_2$ is not violated in $(r_1, 1)$ because $\mathcal{K}_2\,(r_1, 1) \cap (\mathcal{K}_1\,(r_a, m_a) \cup \mathcal{K}_1\,(r_b, m_b)) \neq \emptyset$ with $policy_2(r, m) := \{\{\mathcal{K}_1\,(r_a, m_a), \mathcal{K}_1\,(r_b, m_b)\}\}$
- $policy_3$ again is not preserved because $\mathcal{K}_2\,(r_1, 1) \cap \mathcal{K}_1\,(r_{ab}, m_{ab}) = \emptyset$ with $policy_3(r, m) := \{\{\mathcal{K}_1\,(r_a, m_a), \mathcal{K}_1\,(r_b, m_b)\}, \{\mathcal{K}_1\,(r_{ab}, m_{ab})\}\}$

In the following, we will see that both *total f-secrecy* and *C-secrecy* are special cases of policy-based secrecy each with a particular kind of policy.

The idea of total f-secrecy [3] is that agent j can declare relevant information about his state with a function $f : \mathcal{PT}(\mathcal{R}) \to V$ that only depends on his state. Total f-secrecy requires that for all points $(r, m) \in \mathcal{PT}(\mathcal{R})$ and values v in the range of f, $\mathcal{K}_i(r, m) \cap f^{-1}(v) \neq \emptyset$ (where $f^{-1}(v)$ is simply the preimage of v). The arbitrary set V contains abstract values that denote information which the attacker should not rule out. Next, for total f-secrecy we define the following policy:

$$\mathit{policy}_f(r,m) := \{\{\mathcal{K}_j(r',m') \mid f(r',m') = v\} \mid v \in V\} \tag{1}$$

Note firstly that policy_f is a *global* policy since it is a constant so that policy_f depends on neither r nor m. Secondly, note that because f can be understood as a function on j's state, the sets I_k in policy_f induce a partition of agent j's states. Vice versa a partition of agent j's states induces a function on his states. Hence, as concluded by the following Proposition 1, total f-secrecy corresponds to policy-based secrecy for the class of all global policies where the I_k yield a partition of j's states.

Proposition 1 (Total f-Secrecy Is policy_f-based Secrecy). *Let f be a j-information function and policy_f the resulting policy given by Equation* (1)*. Then agent j maintains total f-secrecy with respect to agent i in $\mathcal{R}$ iff j maintains policy_f-based secrecy with respect to i in $\mathcal{R}$.*

For space reasons, instead of the full proof we give the key to the proof in the following: Because f only depends on agent j's state and in a set $\mathcal{K}_j(r,m)$ agent j has the same local state by definition, all points $(r'',m'') \in \mathcal{K}_j(r',m')$ in the definition of policy_f in (1) have the same evaluation under f, say v. Thus the set $\{K_j(r',m') \mid f(r',m') = v\}$ exactly represents the abstract information v about agent j's state. Both for total f-secrecy and policy_f-based secrecy, the attacker agent i should not rule out this information.

In the same fashion we analyze C-secrecy in terms of policy-based secrecy. The idea of C-secrecy is, that at a point (r,m) agent j can locally allow agent i to exclude some of his states, i.e., all of agent j's states in $\mathcal{PT}(\mathcal{R}) \setminus C(r,m)$. Thus $C(r,m)$ are the only relevant points. C-secrecy requires that for all points $(r,m) \in \mathcal{PT}(\mathcal{R})$ and $(r',m') \in C(r,m)$, we have $\mathcal{K}_i(r,m) \cap \mathcal{K}_j(r',m') \neq \emptyset$. We reformulate C-secrecy with the help of the following j-possibility policy:

$$\mathit{policy}_C(r,m) := \{\{\mathcal{K}_j(r',m')\} \mid (r',m') \in C(r,m)\} \tag{2}$$

Firstly, note that policy_C needs not be global unlike policy_f. Secondly, all sets I_k are unary. A unary I_k just represents a state of agent j. Vice versa, if a policy just contains unary sets I_k in each point (r,m), so that the I_ks cover $\mathcal{K}_i(r,m)$[1], an i-allowability function C can be defined by selecting at least one point from $\mathcal{K}_j(r',m')$ where $I_k = \{\mathcal{K}_j(r',m')\}$ for each $I_k \in \mathit{policy}(r,m)$. As a conclusion, C-secrecy corresponds to policy-based secrecy for the class of all policies, where the I_ks are unary and cover $\mathcal{K}_i(r,m)$ in each point (r,m).

Proposition 2 (C-secrecy Is policy_C-based Secrecy). *Let C be an i-allowability function and policy_C the resulting policy given by Equation* (2)*. Then agent j maintains C-secrecy with respect to agent i in $\mathcal{R}$ iff j maintains policy_C-based secrecy with respect to i in $\mathcal{R}$.*

[1] The covering is required for the i-allowability function C in [3].

This proposition follows almost immediately from the definitions: For C-secrecy the attacker agent i should not rule out $K_j(r', m')$ for each $(r', m') \in C(r, m)$. Each such $K_j(r', m')$ appears in $\mathit{policy}_C(r, m)$ as relevant information $I = \{K_j(r', m')\}$. A formal proof cannot be presented here due to lack of space.

One objective mentioned in the introductory section was to separate different aspects of an information hiding property. In this sense, a policy specifies

1. **what** to protect as pieces of information I_k about j's state
2. **from whom** to protect by targeting a potential adversary agent i the same as with the secrecy notions in [3]
3. **how well** to protect by stating a possibilistic property "agent i considers possible, that" the same as with the possibilistic notions in [3]
4. **when** to protect as a local declaration $\mathit{policy}(r, m)$

In comparison, total f-secrecy is first confined in saying **what** to protect because only certain types of policies are allowed and second in saying **when** to protect because the policy is global. As for C-secrecy, also only certain types of policies are allowed, so that C-secrecy is confined in saying **what** to protect compared to policy-based secrecy. In the next section, we will view the confidentiality requirements of CQE in the light of these considerations.

3 CQE in the Runs & Systems Framework

Controlled Query Evaluation (CQE) is a framework together with several implemented control mechanisms to prevent single users from inferring secret information from disclosed data items. In this section our main goal is to reformulate confidentiality preservation of CQE in the Runs & Systems framework, as motivated in the introduction. In this section we proceed as follows: In Section 3.1 we describe the general components of a CQE control mechanism (a CQE implementation); In Section 3.2 the CQE framework is modeled as a system in Runs & Systems; Section 3.3 discusses our major result of Section 3, namely, confidentiality preservation in the CQE context can be stated in the Runs & Systems model in line with the work of [3] as a special case of policy-based secrecy.

3.1 The Setting

At first we describe a general setting of CQE, into which several implementations like in [10] can be embedded.

- Inst: the set of all database instances
- L_Q: a query language, together with
- $\mathit{eval} : L_Q \times \mathit{Inst} \rightarrow V$: a query evaluation function where $V = \{\mathit{true}, \mathit{false}\}$ or $V = \{\mathit{true}, \mathit{false}, \mathit{undefined}\}$
- L_{ps}: a policy specification language, together with
- $\mathit{eval}_{ps} : L_{ps} \times \mathit{Inst} \rightarrow \{\mathit{true}, \mathit{false}\}$: an evaluation function, that interprets a sentence in L_{ps} in a database instance

– "*User Awareness*": assumptions about what the user knows about the setting or the implementation of CQE or its input

Generally $eval_{ps}(F)(db) = true$ is to be read "F holds in db". Especially, if F expresses prior knowledge or database constraints $eval_{ps}(F)(db) = true$ means that F is consistent with db or that db fulfills F.

Next, we describe the elements of the control mechanism, the implementation *cqe*. In this section, we fix our illustrations on a single user system. Several users would be each treated like a single user. At design time a database administrator sets up the constraints *con* for the database instance, a user administrator models the assumed a priori knowledge $view_{CQE,U}$ of the user about the CQE system as a set of sentences in L_{ps}. The denotation $view_{CQE,U}$ makes clear, that this set is about what the system CQE supposes an user U to know (in the CQE literature $view_{CQE,U}$ is usually denoted as *prior*). The security administrator specifies the information a user should not know as a set *pot_sec* of sentences in L_{ps}, called *potential secrets*. The fact, that a potential secret ψ holds in a database instance db, i.e., $eval_{ps}(\psi)(db) = true$, must be hidden from the user but if $eval_{ps}(\psi)(db) = false$, this fact can be disclosed to the user. The *cqe* implementation employs an *enforcement method*, i.e., *refusal*, *lying* or *combined*. Answers from the database, that leak sensitive information, are either replaced by *mum* (*refusal*), or replaced by a lie (*lying*) or both methods are combined.

Definition 3 (CQE Implementation *cqe***).** *An implementation cqe has the following input:*

– *pot_sec: a confidentiality policy with* $pot\_sec \subseteq L_{ps}$ *(usually finite)*
– *con: database constraints with* $con \subseteq view_{CQE,U}$ *(usually finite)*
– $view_{CQE,U}$*: the user's assumed a priori knowledge with* $view_{CQE,U} \subseteq L_{ps}$ *(usually finite)*
– *db: a database instance with* $db \in Inst$
– $Q := \langle \phi_1, \ldots, \phi_k \rangle$*: a finite sequence of queries with* $\phi_i \in L_Q$

Furthermore an implementation cqe requires preconditions on its input. CQE always demands con to be fulfilled in db. We denote the output by

$$cqe(db, pot\_sec, view_{CQE,U}, Q) = (ans_1, \ldots, ans_k)$$

where $ans_i \in \{true, false\}(\cup\{undefined\})(\cup\{mum\})$ *(the cases in brackets depend on the range of the query evaluation function eval and on the enforcement method). Intermediate computation steps are denoted by*

$$cqe(db, pot\_sec, log_{i-1}, \phi_k) = (ans_i, log_i),$$

where log_i *is the logfile produced at step i with the initial logfile being* $log_0 = view_{CQE,U}$.

Depending on the implementation, the logfile log_i may be used to keep track of the information revealed to the user and to compute the user's inferences from his knowledge. Later we will sketch an implementation of CQE to illustrate the reformulation of CQE in the Runs & Systems framework.

3.2 Describing a CQE System in Runs & Systems

A CQE system can be seen as a system of two agents U, the user, and CQE, the *cqe* algorithm, with U sending queries ϕ to CQE and CQE replying to ϕ. The state of U must encode everything that U observes during system execution or previously knows about the system like the confidentiality policy or database constraints. The state of agent CQE must encode everything that is needed by the *cqe* implementation to run, e.g., the input parameters, the computed logfile etcetera. Next we formally translate all parts of a CQE system into a system $\mathcal{R}_{CQE}$ of the Runs & Systems framework. A global state (s_U, s_{CQE}) is described by $s_U := ([\mathit{pot\_sec},]\mathit{view}_{CQE,U}, \langle \phi_1?\mathit{ans}_1!, \phi_2?\mathit{ans}_2!, \ldots, \phi_k?\mathit{ans}_k!\rangle, \mathit{con})$ and $s_{CQE} := (\mathit{pot\_sec}, \mathit{log}_k, \mathit{db})$ with $\mathit{log}_0 := \mathit{view}_{CQE,U}$. First, the state of agent U might contain the policy depending on whether or not the user is assumed to know the confidentiality policy *pot_sec*. Second, by containing $\mathit{view}_{CQE,U}$ in agent U's state, we claim that agent U knows what a priori knowledge agent CQE assumes about U. Third, the sequence $\langle \phi_1?\mathit{ans}_1!, \phi_2?\mathit{ans}_2!, \ldots\rangle$ are agent U's observations. And last, by containing *con* we claim that agent U knows the database constraints. In a *permissible initial state* of $\mathcal{R}_{CQE}$ the database constraints *con* contained in s_U must hold in *db* and *pot_sec*, $\mathit{view}_{CQE,U}$ and *db* satisfy *precondition* as prescribed by the *cqe* implementation. The policies in agent U's and CQE's states are equal, whenever U knows the confidentiality policy. We now define the runs of $\mathcal{R}_{CQE}$ by simulating the execution of the *cqe* implementation in agent CQE:

Definition 4 (The System $\mathcal{R}_{CQE}$). *Let $Q := \langle \phi_1, \ldots, \phi_n\rangle$ be a finite query sequence in $\mathcal{L}_Q$ and pot_sec, $\mathit{view}_{CQE,U}$ and db an input which satisfies precondition and db satisfies the database constraints included in $\mathit{view}_{CQE,U}$. The run $r_{Q,\mathit{pot\_sec},\mathit{view}_{CQE,U},\mathit{db}}$ starts from the permissible initial state*

$$r_{Q,\mathit{pot\_sec},\mathit{view}_{CQE,U},\mathit{db}}(0) := (([\mathit{pot\_sec},]\mathit{view}_{CQE,U}, \langle\rangle, \mathit{con}), (\mathit{pot\_sec}, \mathit{log}_0, \mathit{db}))$$

where $\mathit{log}_0 := \mathit{view}_{CQE,U}$ and continues as follows for $k \leq n$:

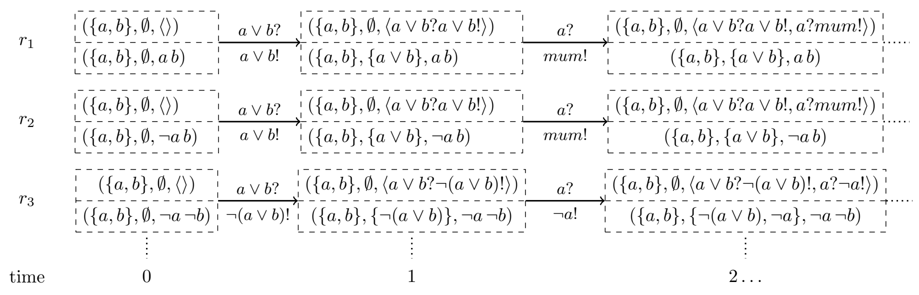
$$\begin{aligned} &r_{Q,\mathit{pot\_sec},\mathit{view}_{CQE,U},\mathit{db}}(k) := (\\ &\quad ([\mathit{pot\_sec},]\mathit{view}_{CQE,U}, \langle \phi_1?\mathit{ans}_1!, \ldots, \phi_k?\mathit{ans}_k!\rangle, \mathit{con}), (\mathit{pot\_sec}, \mathit{log}_k, \mathit{db})) \end{aligned}$$

where $(\mathit{ans}_k, \mathit{log}_k) = \mathit{cqe}(\mathit{db}, \mathit{pot\_sec}, \mathit{log}_{k-1}, \phi_k)$. After all queries in Q the global state does not change any longer:

$$r_{Q,\mathit{policy},\mathit{prior},\mathit{db}}(k) = r_{Q,\mathit{policy},\mathit{prior},\mathit{db}}(n) \text{ where } k > n.$$

The system $\mathcal{R}_{CQE}$ is the set of all possible runs $r_{Q,\mathit{pot\_sec},\mathit{view}_{CQE,U},\mathit{db}}$.

Let $\mathit{db}(r,m)$, $\mathit{pot\_sec}(r,m)$, $\mathit{view}_{CQE,U}(r,m)$ and $\mathit{log}(r,m)$ denote the system components in a point $(r,m) \in \mathcal{PT}(\mathcal{R}_{CQE})$. The definition of $\mathcal{R}_{CQE}$ implicitly contains the assumptions, that the user is aware of the input parameters and the implementation *cqe* (including *precondition*) and knows that the database constraints always hold in an instance. For simplifying the exposition, in addition



r_1
$(\{a,b\},\emptyset,\langle\rangle)$
$(\{a,b\},\emptyset,a\,b)$
$a \vee b?$
$a \vee b!$
$(\{a,b\},\emptyset,\langle a \vee b?a \vee b!\rangle)$
$(\{a,b\},\{a \vee b\},a\,b)$
$a?$
$mum!$
$(\{a,b\},\emptyset,\langle a \vee b?a \vee b!, a?mum!\rangle)$
$(\{a,b\},\{a \vee b\},a\,b)$
r_2
$(\{a,b\},\emptyset,\langle\rangle)$
$(\{a,b\},\emptyset,\neg a\,b)$
$a \vee b?$
$a \vee b!$
$(\{a,b\},\emptyset,\langle a \vee b?a \vee b!\rangle)$
$(\{a,b\},\{a \vee b\},\neg a\,b)$
$a?$
$mum!$
$(\{a,b\},\emptyset,\langle a \vee b?a \vee b!, a?mum!\rangle)$
$(\{a,b\},\{a \vee b\},\neg a\,b)$
r_3
$(\{a,b\},\emptyset,\langle\rangle)$
$(\{a,b\},\emptyset,\neg a\,\neg b)$
$a \vee b?$
$\neg(a \vee b)!$
$(\{a,b\},\emptyset,\langle a \vee b?\neg(a \vee b)!\rangle)$
$(\{a,b\},\{\neg(a \vee b)\},\neg a\,\neg b)$
$a?$
$\neg a!$
$(\{a,b\},\emptyset,\langle a \vee b?\neg(a \vee b)!, a?\neg a!\rangle)$
$(\{a,b\},\{\neg(a \vee b),\neg a\},\neg a\,\neg b)$
time
0
1
2...

Fig. 1. An excerpt of $\mathcal{R}_{CQE}$ with refusal

to Definition 4 we assume that the user knows nothing else about the database, i.e., $view_{CQE,U} = con$.[2] We conclude this section with an implementation of CQE with refusal (taken from [10]) in the Runs & Systems framework, at the same time continuing Example 1 in the CQE context:

Example 3 (CQE with known policy and refusal). We consider propositional databases over the schema with alphabet $\mathcal{D} := \{a, b\}$ and database constraints $con = \emptyset$, i.e., a database instance can be any propositional interpretation of the atomic propositions in $\mathcal{D}$. Queries are expressed in propositional logic over the alphabet $\mathcal{D}$ and so is the confidentiality policy and the user knowledge (in $view_{CQE,U}$ and log). The evaluation functions are $eval(\phi)(db) := db \models \phi$ and $eval_{ps}(\psi)(db) := db \models \psi$ with the propositional model operator $\models$. The user is assumed to know the confidentiality policy. A global state $(s_U, s_{CQE}) = ((pot\_sec, view_{CQE,U}, \langle \phi_1?ans_1!, \dots \rangle, con), (policy, log_k, db))$ is drawn with s_U in the upper and s_{CQE} in the lower part of a state. We omit the part con of s_U in the graphic because we assumed $con = view_{CQE,U}$. On each state transition, the issued query $\phi?$ and the response $ans!$ is denoted. These labels beside the transitions are only for a better readability but not part of the system model in Runs & Systems. For a better understanding of agent CQE's strategy, note that agent CQE keeps the local invariant $log(r, m) \not\models \psi$ for all $\psi \in pot\_sec(r, m)$ (more details can be found in [10]). The graphic in Figure 1 shows an excerpt of the system $\mathcal{R}_{CQE}$ with $pot\_sec = \{a, b\}$, $view_{CQE,U} = con = \emptyset$ and agent U issuing the sequence of queries $\langle a \vee b, a\rangle$.

3.3 Confidentiality Preservation

In the beginning, we consider confidentiality preservation in CQE. The main idea is: whatever the user learns about the database instance db by querying it, he is never sure that a potential secret holds in db.

[2] Generally, the a priori knowledge of U in the Runs & Systems model is encoded in $\mathcal{K}_U(r, 0)$ for each run r. In the CQE literature often $\langle \phi_1?ans_1, \phi_2?ans_2, \dots \rangle$ might be non-empty in an initial state.

Definition 5 (Confidentiality Preservation in CQE). *(see Definition 1 of [10]) An implementation cqe as given by Definition 3 preserves confidentiality iff*

> *for every confidentiality policy pot_sec,*
> *for every initial user knowledge $view_{CQE,U}$ – including database constraints con,*
> *for every database instance $db \in Inst$ that is consistent with con,*
> *so that pot_sec, $view_{CQE,U}$ and db satisfy precondition,*
> *for every finite query sequence Q, for all $\psi \in pot\_sec$*

there exist an alternative database instance *db' and policy pot_sec', so that*

*(a) [*permissible inputs*] pot_sec', $view_{CQE,U}$ and db' satisfy precondition and db' satisfies the database constraints*
*(b) [*same answers*]*
$cqe(db, pot\_sec, view_{CQE,U}, Q) = cqe(db', pot\_sec', view_{CQE,U}, Q)$

and db' refutes ψ

(c) [ψ protected*] $eval_{ps}(\psi)(db') = false$.*

If the user is aware of the confidentiality policy, it holds

*(d) [*same policy*] $pot\_sec' = pot\_sec$.*

In other words: the user can never exclude that a potential secret does *not* hold in *db* (because of the existence of a db'). So the information, that a potential secret ψ does not hold in such an instance $db'(r, m)$, is relevant to ensure the confidentiality of ψ.
Now we formulate an equivalent notion of secrecy for agent CQE in $\mathcal{R}_{CQE}$ using policy-based secrecy:

$$policy_{\mathcal{R}_{CQE}}(r,m) := \{\{\mathcal{K}_{CQE}(r',m') \mid eval_{ps}(\psi)(db(r',m')) = false\} \mid \psi \in pot\_sec(r,m)\} \quad (3)$$

In order to ensure the confidentiality of the property "ψ holds in agent CQE's local database", the complement property "ψ does not hold in agent CQE's local database" must be a target of the CQE-possibility policy $policy_{\mathcal{R}_{CQE}}$. In the remainder of the article I_ψ denotes the set $\{\mathcal{K}_{CQE}(r',m') \mid eval_{ps}(\psi)(db(r',m')) = false\}$. The following theorem shows that we can adequately express the requirements of Definition 5 in the Runs & Systems framework using $policy_{\mathcal{R}_{CQE}}$-based secrecy:

Theorem 1 (Equivalence of Confidentiality Preservation and $policy_{\mathcal{R}_{CQE}}$-based Secrecy). *Given an implementation cqe and the associated system $\mathcal{R}_{CQE}$, agent CQE maintains $policy_{\mathcal{R}_{CQE}}$-based secrecy with respect to agent U according to Definition 2 iff cqe preserves confidentiality according to Definition 5.*

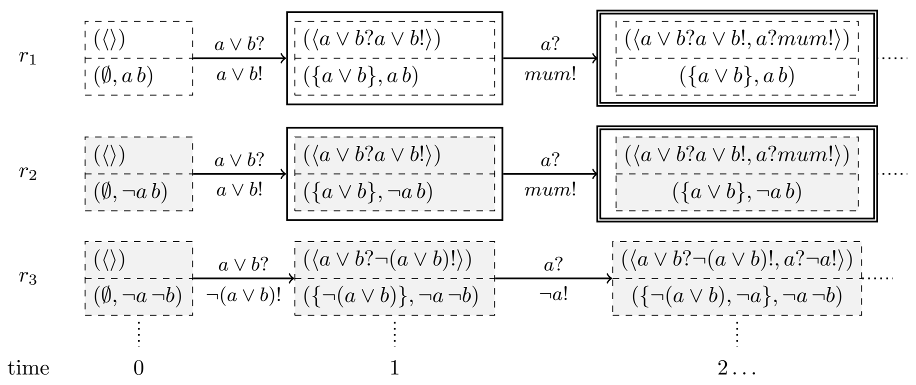
The main keys to the proof are listed in the following.

- The simulation of the *cqe* implementation by agent CQE in the system $\mathcal{R}_{CQE}$:
  especially, only permissible input states are considered (ensuring point (a) of Definition 5).
- The definition of the state of agent U:
  This state encodes the user's prior information, awareness and observations according to the assumptions about the user's abilities. Moreover, the state of agent U determines $\mathcal{K}_U$ in the Runs & Systems framework and models the alternative situations from the user's point of view described by points (b) and (d) of Definition 5.
- The definition of $policy_{\mathcal{R}_{CQE}}$:
  This policy declares the fact that db refutes ψ, i.e., point (c) of Definition 5, as relevant information, which agent U, the user, should not rule out.

The full proof is omitted due to lack of space. Next, we illustrate $policy_{\mathcal{R}_{CQE}}$-based secrecy by continuing Example 3.

Example 4. (CQE with known policy and refusal (continued)): Our illustrations (cf. Figure 2) focus on the protection of the potential secret a in the points $(r_1, 1)$ and $(r_1, 2)$. In the graphics, the state representation is simplified to $(s_U, s_{CQE}) = ((\langle \phi_1?ans_1!, \ldots, \phi_k?ans_k!\rangle), (log_k, db))$ for the sake of readability, yet, agent U knows *con*, $view_{CQE,U}$ and *pot_sec*. In run r_1 at time 1 agent U, when reasoning about the possible system states, might determine $\mathcal{K}_U(r_1, 1)$, combining all "run-time" information available to him (encoded in s_U) and the knowledge about the system (the specification $\mathcal{R}_{CQE}$). The relevant information for the protection of the potential secret a is $I_a = \{\mathcal{K}_{CQE}(r, m) \mid eval_{ps}(a)(db(r, m)) = false\}$. In both $(r_1, 1)$ and $(r_1, 2)$ the condition of $policy_{\mathcal{R}_{CQE}}$-based secrecy is fulfilled (with "witnesses" $(r_2, 1)$ and $(r_2, 2)$ respectively) for $I_a \in policy(r_1, 1)$ and $I_a \in policy(r_1, 2)$ respectively.
You can observe that, if in r_2 at time 2 agent CQE returned the correct answer $\neg a!$ instead of *mum*!, agent U would have learned at point $(r_1, 2)$ that a holds in the current database instance and at point $(r_2, 2)$ that b holds in the current database instance. In run r_2, the disclosure of b to agent U is prohibited by the local confidentiality policy. The first harm of the local confidentiality policy would not be detected by means of the logfile $log(r_1, 2)$, i.e., $log(r_1, 2) \not\models a$ and thus is called *meta-inference* ($\models$ denotes logical implication in propositional logic). The second inference would be detected by means of the logfile $log'(r_2, 2) = log(r_2, 2) \cup \{\neg a\} \models b$. In this situation, we notice that only by means of the logfile $log(r, m)$ and logical implication $\models$, agent CQE cannot determine $\mathcal{K}_U(r, m)$, as we have seen, that agent CQE cannot detect all inferences of agent U with the logfile. But agent CQE's *strategy*, i.e., the strategy of the algorithm *cqe*, nevertheless prevents such harmful inferences by additional refusals.

Unlike policy-based secrecy, total f-secrecy and C-secrecy do not suffice to express the confidentiality requirement in the CQE context.



r_1
$(\langle\rangle)$
$(\emptyset, a\,b)$
$a \vee b?$
$a \vee b!$
$(\langle a \vee b?a \vee b!\rangle)$
$(\{a \vee b\}, a\,b)$
$a?$
$mum!$
$(\langle a \vee b?a \vee b!, a?mum!\rangle)$
$(\{a \vee b\}, a\,b)$
r_2
$(\langle\rangle)$
$(\emptyset, \neg a\,b)$
$a \vee b?$
$a \vee b!$
$(\langle a \vee b?a \vee b!\rangle)$
$(\{a \vee b\}, \neg a\,b)$
$a?$
$mum!$
$(\langle a \vee b?a \vee b!, a?mum!\rangle)$
$(\{a \vee b\}, \neg a\,b)$
r_3
$(\langle\rangle)$
$(\emptyset, \neg a\,\neg b)$
$a \vee b?$
$\neg(a \vee b)!$
$(\langle a \vee b?\neg(a \vee b)!\rangle)$
$(\{\neg(a \vee b)\}, \neg a\,\neg b)$
$a?$
$\neg a!$
$(\langle a \vee b?\neg(a \vee b)!, a?\neg a!\rangle)$
$(\{\neg(a \vee b), \neg a\}, \neg a\,\neg b)$
time
0
1
2...

$prior = con = \emptyset \qquad pot\_sec = \{a, b\}$

Fig. 2. CQE with known policy and refusal: Agent U considers possible in $(r_1, 1)$ and $(r_1, 2)$, that a does not hold, which is visualized by the non-empty intersection of the gray colored $(I_a = \{\mathcal{K}_{CQE}(r, m) \mid eval_{ps}(a)(db(r, m)) = \mathit{false}\})$ with the points with a single frame $(\mathcal{K}_U(r_1, 1))$ or with the points with a double frame resp. $(\mathcal{K}_U(r_1, 2))$.

Proposition 3 (Confidentiality, Total f-Secrecy and C-Secrecy). *Given an implementation cqe for database instances Inst $\supseteq \{(), (a), (b), (a, b)\}$ like in Example 1 and the associated system $\mathcal{R}_{CQE}$, there is no function $f : \mathcal{PT}(\mathcal{R}_{CQE}) \to V$, such that cqe satisfies the requirements of Definition 5 iff agent CQE maintains total f-secrecy with respect to agent U in $\mathcal{R}_{CQE}$. Also there is no function $C : \mathcal{PT}(\mathcal{R}_{CQE}) \to \mathcal{P}(\mathcal{PT}(\mathcal{R}_{CQE}))$, such that cqe satisfies the requirements of Definition 5 iff agent CQE maintains C-secrecy with respect to agent U in $\mathcal{R}_{CQE}$.*

In our proof sketch we consider a policy $pot\_sec = \{a, b\}$ as an input to the *cqe* implementation. This policy can be equivalently enforced in $\mathcal{R}_{CQE}$ in a point $(\tilde{r}, 0)$ by the local policy $policy(\tilde{r}, 0) = \{I_a, I_b\}$ with $I_a = \{\mathcal{K}_{CQE}(r, m) \mid eval_{ps}(a)(db(r, m)) = \mathit{false}\}$ and I_b analogously (Theorem 1). Policy-based secrecy for *policy* can neither be enforced by the means of total f-secrecy nor by the means of C-secrecy:

– We cannot find a function f with

$$\begin{aligned} &policy_f(\tilde{r}, 0) \\ &= \{\{\mathcal{K}_{CQE}(r', m') \mid f(r', m') = v_1\}, \{\mathcal{K}_{CQE}(r', m') \mid f(r', m') = v_2\}\} \\ &= \{I_a, I_b\} = policy(\tilde{r}, 0) \text{ by (1)}, \end{aligned}$$

because for a $(r', m') \in I_a \cap I_b (\neq \emptyset)$ either $f(r', m') = v1 \neq v_2$ or $f(r', m') = v_2$. Hence, by Proposition 1, *policy*-based secrecy (and the requirements of Definition 5 for *pot_sec*) cannot equivalently be enforced by total f-secrecy.

– I_a contains at least the two different information sets $\mathcal{K}_{CQE}(r_1, m_1)$ with $eval(b)(db(r_1, m_1)) = false$ and $\mathcal{K}_{CQE}(r_2, m_2)$ with $eval(b)(db(r_2, m_2)) = true$ where both $eval_{ps}(a)(db(r_1, m_1)) = eval_{ps}(a)(db(r_2, m_2)) = false$. Hence we cannot find a function C with

$$\begin{aligned} policy_C(\tilde{r}, 0) &= \{\{\mathcal{K}_{CQE}(r', m')\}, \{\mathcal{K}_{CQE}(r'', m'')\}\} \\ &= \{I_a, I_b\} = policy(\tilde{r}, 0) \text{ by (2)} \end{aligned}$$

and, by Proposition 2, *policy*-based secrecy (and Definition 5) cannot equivalently be enforced by C-secrecy.

4 Policy-Based Secrecy in Interpreted Systems

Halpern and O'Neill in [3] also use a *logic of knowledge and time* (introduced in [4]) to express properties of a system. Several of their notions of secrecy can be equivalently described in terms of this logic, i.e. syntactically. Next, we recapitulate this logic in brief: A formula F is build by $F :: \equiv A | F \vee F | F \wedge F | K_i\, F | \Diamond F$ where A is an atomic proposition from an alphabet $\mathcal{A}$. The meaning of a formula F is given by an interpreted system:

Definition 6 (Interpreted System $\mathcal{I}$). *(Section 2 of [3]) Let $\mathcal{R}$ be a system and $\mathcal{A}$ an alphabet of atomic propositions. An interpretation π of $\mathcal{A}$ in $\mathcal{R}$ is a function*

$$\pi : \mathcal{A} \times \Sigma \to \{true, false\}$$

where Σ is the set of all global states in $\mathcal{R}$. The pair $(\mathcal{R}, \pi)$ is called an interpreted system and denoted by $\mathcal{I}$.

The interpretation of the connectives is as usual and the modal operators are interpreted as follows:

$$(\mathcal{I}, r, m) \models K_i\, \varphi \text{ iff } (\mathcal{I}, r', m') \models \varphi \text{ for all } (r', m') \in \mathcal{K}_i(r, m)$$

$$(\mathcal{I}, r, m) \models \Diamond \varphi \text{ iff there exists an } n \text{ such that } (\mathcal{I}, r, n) \models \varphi$$

A formula whose evaluation depends only on the state of an agent j in $\mathcal{I}$ is called *j-local*, formally: For all points (r, m) and (r', m') such that $r_j(m) = r'_j(m')$ it holds that $(\mathcal{I}, r, m) \models \varphi$ iff $(\mathcal{I}, r', m') \models \varphi$.

Like Halpern and O'Neill, we restate policy-based secrecy in terms of this logic:

Proposition 4 (Policy-Based Secrecy in Interpreted Systems)). *Let $\mathcal{R}$ be a system, $\mathcal{A}$ a propositional alphabet and policy a j-possibility policy. Agent j maintains policy-based secrecy in $\mathcal{R}$ with respect to agent i iff for all interpretations π and points $(r, m) \in \mathcal{PT}(\mathcal{R})$:*
if a formula φ is j-local in $\mathcal{I} = (\mathcal{R}, \pi)$ and for some $I_k \in policy(r, m)$:

$$(I, r', m') \models \varphi \text{ iff } (r', m') \in \bigcup_{S \in I_k} S \tag{4}$$

then $(\mathcal{I}, r, m) \models \neg K_i \neg \varphi$.

Let us review Proposition 4: [3] offers several interpretations of j-locality among which the following is useful in our case: A j-local formula can be understood as a property of agent's j state. So Equation (4) says that regarding π the formula φ expresses the same property as I_k does. Now the equivalent criterion for *policy*-based secrecy reads as: whenever a j-local formula φ describes relevant information about j's state according to $policy(r, m)$, agent i cannot rule out that φ holds, i.e., agent i doesn't know that φ does not hold. The proof cannot be shown here due to lack of space. This syntactic characterization gives ideas how a notion of secrecy can be slightly adapted to suit other purposes (cf. [8]), e.g.,

– $(\mathcal{I}_{CQE}\ , r, m) \models \neg K_U \Diamond\psi$: agent U does not know, whether ψ held, holds or will hold at some time during system execution
– $(\mathcal{I}_{CQE}\ , r, m) \models C \rightarrow \neg K_U\, \psi$: only if condition C is satisfied, the potential secret ψ must be protected

5 Related Work

In [11] Hughes and Shmatikov propose a "technique (which) combines the benefits of the knowledge based approach (e.g., like in this article), namely, natural specification of information hiding properties, with those of the process algebra-based approach, namely, natural specification of system behavior." The confidential data items are attributes which are functions each with a type $X \rightarrow Y$. E.g., $bal : X \rightarrow \mathbb{R}$ returns the balance of a customer x's ($\in X$) bank account. Such an attribute is instantiated in each system state with a function of type $X \rightarrow Y$, e.g., bal_C : Jones $\mapsto$ 10000, Smith $\mapsto$ 36000 with $X = \{\text{Smith}, \text{Jones}\}$. An attacker can acquire partial knowledge of a function, e.g., an attribute, which is represented by a *function view* $N = \langle F, I, K\rangle$. A function view comprises the attacker's knowledge about the graph, e.g., $f(x) \neq y$, by $f \subseteq F \subseteq X \times Y$, about the range of the image, e.g., $y \in f(X)$, by $I \subseteq f(X)$, and about the kernel ker, e.g., $f(x) = f(x')$, by an equivalence relation K on $X \times X$. In the first example, knowing F and thus $(x, y) \notin F$ the attacker knows that the point (x, y) does not lie on the graph of f, i.e., $f(x) \neq y$. The semantics of a function view N is given as $lin(N) := \{f : X \rightarrow Y \mid f \subseteq F, I \subseteq f(X), K \subseteq ker\, f\}$ which is the set of functions an attacker with knowledge N considers possible. Additionally, a function view N of f is consistent with f, i.e., $f \in lin(N)$, and *closed*, which means it accounts for all inferences an attacker can draw due to the interrelations between F, I and K. Hughes and Shmatikov express information hiding properties in terms of function views, e.g., *Z-value opaqueness* which requires that the attacker with view N of f considers possible that f might output any value in Z on an arbitrary input x. As an application of function views, one can declare opaqueness properties for attributes with respect to the view of an attacker. This view of the attacker in a system state C is the function view representing $L^{\sim}_{\alpha,C} = \{\alpha_{C'} : X \rightarrow Y \mid C \sim C'\}$. Thereby, the equivalence relation $C \sim C'$ on system states denotes that the attacker cannot distinguish the states C and C'.

In our context of policy-based secrecy, a system has the states $\mathcal{PT}(\mathcal{R})$ with the equivalence relation $\mathcal{K}_i$ for the attacker agent i. All attributes of interest are the relevant properties I_k of agent j's state. Let $hasPropertyI_k$ denote such an attribute with type $\rightarrow \{0, 1\}$ (i.e., a binary constant), which is instantiated in each state $(r, m) \in \mathcal{PT}(\mathcal{R})$ as a function

$$hasPropertyI_{k(r,m)} = \text{if } \left((r, m) \in \bigcup_{S \in I_k} S \right) \text{ then 1 else 0}$$

Whenever in a state (r, m) a property is declared security-relevant in the local j-possibility policy, the attacker agent i must think that agent j might have this property. Hence, we require $\{1\}$-value opaqueness on agent i's view in (r, m). As a conclusion, we see that policy-based secrecy and hence confidentiality preservation in CQE translates to opaqueness properties of function views as proposed in [11] in a natural way. Thus, CQE might in the future also benefit from the solution of [11] to provide for a natural system specification as well as security property specification for formal verification.

Mantel in [12] formalizes information flow properties in trace-based systems in MAKS (modular assembly kit for security properties) in which he characterized several common information flow properties. Generally, as noted by [13], these information-flow properties might be too restrictive to be useful in the database context. For instance, non-interference [7] does not allow the attacker to conclude that some but not which secret holds in the database instance. Furthermore, as elaborated in [3], separability [5] and generalized non-interference [6] are stricter than synchronous secrecy which is a special case of C-secrecy and total f-secrecy, respectively, in that the former properties model that an agent can observe and reason about an entire *infinite* run. In contrast, C-secrecy etc. and also policy-based secrecy considers an agent to make observations only after finite time. Nevertheless, Mantel's proposal allows for weakened definitions of information flow properties which, for instance, can also be used for hiding the value of an agent's internal variables from another agent while exchanging messages in MAS [14].

Altogether, by choosing the general framework of Halpern and O'Neill we provide a basis for comparing confidentiality requirements of CQE with other approaches in the future. In the discussion of [11] above, we could identify the proper equivalence relation, attributes and opaqueness property to enforce the confidentiality requirements of CQE easily because we expressed the requirements of CQE by policy-based secrecy.

6 Conclusion and Future Work

In this paper, we demonstrated that the notion of confidentiality preservation of CQE complements Halpern and O'Neill's notions of total secrecy, C-secrecy and total f-secrecy in that an agent can declare individually which of the agent's

properties are relevant. In particular, total secrecy being a special case of C-secrecy [3], C-secrecy and total f-secrecy can be enforced by appropriate declarations. Introducing the generic notion of policy-based secrecy we pointed out the essence of confidentiality preservation in CQE in comparison to total f-secrecy and C-secrecy. In particular, we saw that confidentiality preservation in CQE is not expressible by either of these notions. For future work, the results of this paper can be applied to CQE under several aspects:
Time-Related Confidentiality Requirements. The logic of knowledge and time, used to express policy-based secrecy in Proposition 4, offers means to change the requirement of policy-based secrecy in an intuitive way: Consider the following confidentiality requirements of employee Smith where the temporal operator $\triangleleft$ should be read as "in the past":

1. $\neg K_{Jones}\, phone(Smith, 6789)$
2. $\neg K_{Jones}\, \Diamond cancer(Smith)$
3. $\neg K_{Jones}\, [\triangleleft HR(Smith, ProjectA, teamleader) \wedge$
   $\neg HR(Smith, ProjectA, teamleader) \wedge HR(Smith, ProjectA, programmer)]$

Requirement 1) complies with the syntax used in Proposition 4 and expresses that Jones must not know whether Smith *currently* has the phone number 6789. Requirement 2) expresses that Jones must not know whether Smith *ever* had cancer (or has or will have). Even more complex, requirement 3) means that Jones must not know whether Smith has lost his position as a team leader in ProjectA and now works as a programmer. Such time-related requirements make sense in the context of database updates or update operations in MAS. Whereas requirement 1) expresses the requirement of Definition 5, which has been shown in this paper, requirement 2) may be found in the context of controlled update transactions in [1]; and more complex requirements like 3) have not yet been formulated in CQE. Since the formula $\Diamond cancer(Smith)$ is generally not j-local, requirement 2) (as well as 3)) cannot be expressed by policy-based secrecy (cf. Proposition 4), but another semantic representation has to be found.
Attacker Model. Policy-based secrecy assumes that the attacker has full awareness of the system specification and can determine all possible system states at runtime. These assumptions are inherent in the K-operator used in Proposition 4. In Halpern and O'Neill's framework other attacker models have been studied as well using the following inference models: $Pr_{Jones}(cancer(Smith))$ denotes Jones' reasoning about the probability of the fact that Smith has cancer [3]; $X_{Jones}(cancer(Smith))$ denotes Jones' computation of the fact that Smith has cancer [15] (resource-bounded reasoning). Using the algorithmic knowledge operator X_j one can individually model the computational capabilities of the attacker j as demonstrated in [15]. Halpern and O'Neill in [3] discuss more attacker models in their framework. The results presented here can be a starting point to apply these attacker models to CQE.
MAS. Employing the formal framework presented in this article we can formulate – even complex and time-related – confidentiality requirements more intuitively then for example in [10, 1, 9], as suggested in the previous two paragraphs. Yet, we might identify the requirements in [10, 1] with $\neg K_U \psi$ and $\neg K_U \Diamond \psi$ where ψ

doesn't contain temporal operators such as in the examples with employee Smith above. Thus, the results in [10, 1] should be useful to develop protocols for general MAS even to enforce more complex temporal confidentiality requirements.

References

1. Biskup, J., Gogolin, C., Seiler, J., Weibert, T.: Requirements and protocols for inference-proof interactions in information systems. In: Fourteenth European Symposium on Research in Computer Security (ESORICS 2009). Volume 5789 of LNCS. (2009) 285–302
2. Biskup, J.: Usability confinement of server reactions: Maintaining inference-proof client views by controlled interaction execution. In: Sixth International Workshop on Databases in Networked Information Systems (DNIS 2010). Volume 5999 of LNCS. (2010) 80–106
3. Halpern, J.Y., O'Neill, K.R.: Secrecy in multiagent systems. ACM Transactions on Information and System Security **12**(1) (2008)
4. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
5. McLean, J.: A general theory of composition for trace sets closed under selective interleaving functions. In: Fifteenth IEEE Symposium on Security and Privacy (SP 1994). (1994) 79–93
6. McCullough, D.: Specifications for multi-level security and a hook-up property. In: Eighth IEEE Symposium on Security and Privacy (SP 1987). (1987) 161–166
7. Lee, E., Zakinthinos, A.: A general theory of security properties. In: Eighteenth IEEE Symposium on Security and Privacy (SP 1997). (1997) 94–102
8. Halpern, J.Y., O'Neill, K.R.: Anonymity and information hiding in multiagent systems. Journal of Computer Security **13**(3) (2005) 483–512
9. Biskup, J., Kern-Isberner, G., Thimm, M.: Towards enforcement of confidentiality in agent interactions. In: Twelfth International Workshop on Non-Monotonic reasoning (NMR 2008). (September 2008) 179–188
10. Biskup, J., Bonatti, P.: Controlled query evaluation for enforcing confidentiality in complete information systems. International Journal of Information Security **3** (2004) 14–27
11. Hughes, D., Shmatikov, V.: Information hiding, anonymity and privacy: a modular approach. Journal of Computer Security **12**(1) (2004) 3–36
12. Mantel, H.: A uniform framework for the formal specification and verification of information flow security. PhD thesis, Universität des Saarlandes (2003)
13. Cuppens, F., Trouessin, G.: Information flow controls vs interference controls: An integrated approach. In: Third European Symposium on Research in Computer Security (ESORICS 1994). Volume 875 of LNCS. (1994) 447–468
14. Schairer, A.: Towards using possibilistic information flow control to design secure multiagent systems. In: First International Conference on Security in Pervasive Computing (SPC 2003). Volume 2802 of LNCS. (2003) 101–115
15. Halpern, J.Y., Pucella, R.: Modeling adversaries in a logic for security protocol analysis. The Computing Research Repository CoRR **abs/cs/0607146** (2006)