Confidentiality Policies and Their Enforcement for Controlled Query Evaluation

Joachim Biskup[1] and Piero Bonatti[2]

[1] Fachbereich Informatik, Universität Dortmund
D-44221 Dortmund, Germany
biskup@ls6.informatik.uni-dortmund.de
[2] Dipartimento di Tecnologie dell'Informazione, Università di Milano
I-26013 Crema, Italy
bonatti@dti.unimi.it

Abstract. An important goal of security in information systems is *confidentiality*. A confidentiality policy specifies which users should be forbidden to acquire what kind of information, and a controlled query evaluation should enforce such a policy even if users are able to reason about a priori knowledge and the answers to previous queries. We put the following aspects into a unifying and comprehensive framework: formal models of confidentiality *policies* based on potential secrets or secrecies, user *awareness* of the policy instance, and *enforcement* methods applying either lying or refusal, or a combination of lying and refusal. Two new evaluation methods are introduced. Different approaches are systematically compared and evaluated.

Keywords: Inference control; Controlled query evaluation; Confidentiality; Policy; Potential secret; Secrecy; Refusal; Lying; Combined refusal and lying.

1 Introduction

An important goal of security in information systems is *confidentiality*, i.e., the ability to hide specific information from certain users according to some confidentiality policy. Roughly speaking, such a confidentiality policy specifies which users should be forbidden to acquire what kind of information. Typically, a confidentiality policy is expressed by the owner or a designated administrator of the information to be hidden. Using a logic-oriented model of information systems, including the relational model and Datalog, a controlled query evaluation has to enforce such a policy, even if users are able to *infer* more information than what is explicitly returned as an answer to their queries.

In this paper, we deal with the theoretical foundations of controlled query evaluation. Our work is based on a simple but powerful logic-oriented data model which considers an instance of an information system as a structure (interpretation) in the sense of logic, a query as a sentence, and the ordinary answer as the truth value of the query w.r.t. the instance.

D. Gollmann et al. (Eds.): ESORICS 2002, LNCS 2502, pp. 39–55, 2002.

This theoretical model is nonetheless relevant for practical and static approaches to information flow control and inference control [6, 5], including discretionary and mandatory access control, and data customization such as statistical perturbation and polyinstantiation.

Previous work on controlled query evaluation [1, 2, 3, 4, 9] has identified three important aspects:

1. Formal models of confidentiality *policies*: potential secrets versus secrecies.
   Under the model of *secrecies*, a confidentiality policy requires that, given a set of sentences that constitute the policy instance, a user should not be able to infer the truth value of those sentences in the current instance of the information system. Whereas under the model of *potential secrets*, a confidentiality policy requires that, for a given set of sentences, if any of these sentences is true in the current instance of the information system then the user should not be able to infer that fact. However, users are allowed to believe the opposite truth value. Thus under the former model, query answers should not suggest any truth value at all for the given set of sentences, while under the latter model, a predetermined alternative of the truth values can be suggested.
   Clearly, the goal of confidentiality has to be complemented by availability. In this paper, like in previous work, we deal with availability only by observing the following informal heuristic. A controlled query evaluation should be as cooperative to the user as possible, i.e., it should return a distorted answer (i.e., a refusal or a lie) only if necessary for achieving confidentiality.
2. User *awareness* of the policy instance: unknown versus known.
   A user who *knows* the policy instance might use such knowledge to infer hidden information, say, by the following kind of reasoning: "I received a distorted answer. The system did so only because otherwise the correct answer together with previously observed data would logically imply a specific sentence Ψ contained in the policy instance. So, observing the distortion, I conclude that Ψ is valid indeed." Even if the actual policy instance is *unknown* to him, the user might base a similar reasoning solely on his awareness of the model of the policy. Surely, any enforcement method should make such arguments impossible.
3. *Enforcement* method: lying versus refusal, and combined lying and refusal.
   (Uniform) *refusal* returns a `mum`, i.e., it refuses to return the truth value of the query, if the correct answer lets the user infer (either directly or indirectly) one of the sentences of the policy instance. Under similar conditions, (uniform) *lying* returns a lie, i.e., the negation of the correct answer. *Combined* methods are allowed to protect the sentences of the policy instance by both kinds of answer distortion.

Sicherman/de Jonge/van de Riet [9] introduce the topic of controlled query evaluation and discuss refusal for unknown and known secrecies. Bonatti/Kraus/Subrahmanian [4] pioneer lying for known potential secrets. Biskup [1] defines a common framework for studying and comparing enforcement methods dealing

Table 1. Aspects for controlled query evaluation and contributions of previous work

		confidentiality policy			
		potential secrets		secrecies	
enforcement	lying		[4] [2, section 4]	"not applicable"	[1, section 5]
	refusal		[2, section 3]	[9] [1, Section 4.2]	[9] [1, section 4.1]
	combined		[3, section 3]	[3, section 4]	
		unknown	known		unknown
		user awareness			

with lying and refusal for unknown and known secrecies. A main conclusion is that "for unknown secrecies refusal is better than lying". Biskup/Bonatti [2] adapt that framework in order to treat lying and refusal for known potential secrets. They show that in this context lying and refusal are incomparable in general, but functionally equivalent under some natural restriction (i.e., the policy instance should be closed under disjunction). Finally, Biskup/Bonatti [3] suggest a combination of refusal and lying for known policies, including both potential secrets and secrecies. The combined methods avoid the drawbacks of the uniform methods (the mandatory protection of disjunctions for lying, and, for refusal, the independence from the actual instance of the information system).

Table 1 gives a structured overview of all these works, highlighting the combinations of the three major aspects that have not yet been investigated. In this paper, we try to fill in the table's gaps, and provide a unifying view of all these approaches, including a general analysis of their mutual relationships. More precisely, the structure and the contributions of the paper are as follows.

- In Section 2 we introduce a uniform reformulation of the existing approaches, including a unified notion of confidentiality, parametric w.r.t the three classification dimensions.
  Furthermore, this section points out the mutual relationships between the existing approaches for known policies and their counterparts for unknown policies, and the relationships between the methods based on secrecies and their counterparts for potential secrets. In the latter case, we observe that each method based on a policy instance presented as a set of secrecies

$$secr := \{\{\Psi_1, \neg\Psi_1\}, \ldots, \{\Psi_k, \neg\Psi_k\}\} \tag{1}$$

  is equivalent to the corresponding method based on potential secrets, applied to the policy instance

$$pot\_sec(secr) := \{\Psi_1, \neg\Psi_1, \ldots, \Psi_k, \neg\Psi_k\}\,. \tag{2}$$

- In Section 3, we fill in the two table slots corresponding to lies and refusal for unknown potential secrets. The new methods are derived from their coun-

terparts for unknown secrecies by analogy with the reduction of secrecies to potential secrets illustrated in (2).
Then, in the same section, we try to derive a combined method for unknown potential secrets from the corresponding method for known potential secrets, by analogy with the relationships observed in Section 2. Surprisingly, all the natural attempts along this direction fail to preserve confidentiality.

– In Section 4 we prove some general results that explain the relationships empirically observed in Section 2. As a byproduct, we identify some general properties of the controlled query evaluation methods under which the reduction of secrecies to potential secrets based on (2) preserves confidentiality. These results have a potential practical impact on the design of uniform access control mechanisms that support simultaneously both of the policy models (and combinations thereof).

The paper is closed by a section with a discussion of the results and some conclusions. Proofs will be omitted due to space limitations, with the exception of the proof of Theorem 4.

2 A Unified Formal Model for Confidentiality Policies

2.1 Ordinary Query Evaluation

An *information system* maintains two kinds of data: A *schema* DS captures the universe of discourse for the intended application and is formally defined as the set of all allowed instances. An *instance* db is a *structure* which *interprets* the symbols of some logic, i.e. of the universe of discourse (see e.g. [8, 7]). We only consider the most elementary kind of query, namely a sentence in the language of the logic. Given a structure db (stored as an instance) and a sentence Φ (issued as a query), Φ is either *true* (*valid*) or *false* in db, or in other words, the structure is either a *model* of the sentence or not. When a user issues a query Φ against the schema DS, the (ordinary) *query evaluation* $eval(\Phi)$ determines the pertinent case for the current instance db. Thus we formally define

$$eval(\Phi) : DS \to \{true, false\} \text{ with } eval(\Phi)(db) := db \texttt{ model\_of } \Phi\,, \qquad (3)$$

where the boolean operator `model_of` is assumed to be appropriately specified for the logic under consideration.[1] We also use an equivalent formalization where either the queried sentence or its negation is returned:

$$eval^*(\Phi) : DS \to \{\Phi, \neg\Phi\} \text{ with}$$

$$eval^*(\Phi)(db) := \texttt{if } db \texttt{ model\_of } \Phi \texttt{ then } \Phi \texttt{ else } \neg\Phi\,. \qquad (4)$$

[1] This abstract formulation makes our approach compatible with a variety of database models. To get a more concrete understanding of our framework, the reader may instantiate `model_of` with standard first-order satisfaction.

The symbols of the logic comprise both the negation symbol (as implicitely assumed above) and disjunction. We assume that both connectives have their classical semantics . Our definition trivially implies that for all instances db and for all queries Φ we have db `model_of` $eval^*(\Phi)(db)$.

We also define the semantic relationship $\models$ for logical implication in a standard way: $\Phi \models \Psi$ iff for every structure db such that db `model_of` Φ we also have db `model_of` Ψ. The complementary relationship is denoted with $\not\models$.

2.2 Controlled Query Evaluation

Controlled query evaluation consists of two steps. First, the correct answer is judged by some *censor* and then, depending on the output of the censor, some *modificator* is applied. In order to assist the censor, the system maintains a *user log*, denoted by *log*, which represents the explicit part of the *user's assumed knowledge*. Formally, *log* is declared to be a set of sentences. The log is meant to contain all the sentences that the user is assumed to hold true in the instance, in particular publicly known *semantic constraints*. Additionally, the log records the sentences returned as *answers to previous queries*.

Formally we will describe an approach to *controlled query evaluation* by a family of (possibly) partial functions $control\_eval(Q, log_0)$, each of which has two parameters: a (possibly infinite) query sequence $Q = \langle \Phi_1, \Phi_2, \ldots, \Phi_i, \ldots \rangle$, and an initial user log log_0. The inputs to any such function are "admissible" pairs (db, *policy*) where db is an instance of the information system, and *policy* is an instance of a suitably formalized confidentiality policy. The admissibility of an argument pair (db, *policy*) is determined by some formal *precondition* associated with the function. Throughout the paper we suppose that for each function, the policies in its admissible pairs all belong to exactly one of the policy models. The function returns an answer sequence to the user, and updates the user log as a side effect. For any specific function, we indicate the underlying choices w.r.t. the three aspects—model of *p*olicy, user *a*wareness, *e*nforcement method—by a superscript p, a, e with $p \in \{\mathtt{sec}, \mathtt{ps}\}$, $a \in \{\mathtt{unknown}, \mathtt{known}\}$, and $e \in \{L(ying), R(efusal), C(ombined)\}$. In symbols,

$$control\_eval^{p,a,e}(Q, log_0)(db, policy) = \\ \langle (ans_1, log_1), (ans_2, log_2), \ldots, (ans_i, log_i), \ldots \rangle ,$$

where the side effect on the user log is described by

$$log_i := \mathtt{if}\ \ ans_i = \mathtt{mum}\ \mathtt{then}\ \ log_{i-1}\ \ \mathtt{else}\ \ log_{i-1} \cup \{ans_i\} ,$$

and a $censor^{p,a,e}$ is formally described by a truth function (or two truth functions for combined methods) with arguments of the form (Ψ, log, db, *policy*). The censor returns *true* iff the modification e is required.

2.3 Confidentiality Requirements

The syntactical appearance of an *instance* of a confidentiality policy depends on the model: Either a policy instance is given by a finite set *secr* of complementary

pairs of sentences, $secr = \{\{\Psi_1, \neg\Psi_1\}, \ldots, \{\Psi_k, \neg\Psi_k\}\}$, where each pair is called a *secrecy*. Or a policy instance is given by a finite set *pot_sec* of sentences, $pot\_sec = \{\Psi_1, \ldots, \Psi_k\}$, where each sentence is called a *potential secret*.

The original motivations for the two models of confidentiality policies are quite different (see [1, 2] for a detailed exposition).

- A secrecy $\{\Psi, \neg\Psi\}$ specifies the following: A user, seeing only the apriori knowledge log_0 and the returned answer sequence, should not be able to distinguish whether Ψ or $\neg\Psi$ is true in the actual instance of the information system, or speaking otherwise, both cases should be possible for him. More formally, the controlled query evaluation must be "nowhere injective with respect to the secrecy $\{\Psi, \neg\Psi\}$".
- A potential secret Ψ specifies the following: Such a user should not be able to exclude that $\neg\Psi$ is true in the actual instance of the information system, or speaking otherwise, this case should be possible for him. More formally, the controlled query evaluation must have a "surjective restriction on the false potential secret, i.e. on $\neg\Psi$".

The intended semantics is formalized as follows.

Definition 1. *Let* $control\_eval^{p,a,e}(Q, log_0)$ *describe a specific controlled query evaluation with precond as associated precondition for "admissible" arguments, and* $policy_1$ *be a policy instance.*

1. $control\_eval^{p,a,e}(Q, log_0)$ *is defined to* preserve confidentiality[2] *with respect to* $policy_1$ *iff*
   for all finite prefixes Q' *of* Q,
   for all instances db_1 *of the information system such that* $(db_1, policy_1)$ *satisfies precond,*
   and for all $\Theta \in policy_1$,
   there exists db_2 *and* $policy_2$ *such that* $(db_2, policy_2)$ *also satisfies precond and such that the following properties hold:*

   (a) [same answers]

   $control\_eval^{p,a,e}(Q', log_0)(db_1, policy_1) = control\_eval^{p,a,e}(Q', log_0)(db_2, policy_2)$;

   (b) [different secrets/false potential secrets]

   if $p = \texttt{sec}$, i.e., $\Theta = \{\Psi, \neg\Psi\}$ is a secrecy: $\{eval^*(\Psi)(db_1), eval^*(\Psi)(db_2)\} = \{\Psi, \neg\Psi\}$,

   if $p = \texttt{ps}$, i.e., $\Theta = \Psi$ is a potential secret: $eval^*(\Psi)(db_2) = \neg\Psi$;

   (c) [awareness] if $a = \texttt{known} : policy_1 = policy_2$.

2. *More generally,* $control\_eval(Q, log_0)^{p,a,e}$ *is defined to* preserve confidentiality *iff it preserves confidentiality with respect to all "admissible" policy instances.*

[2] Or to be *secure*, as a shorthand used in previous work

The above definition generalizes and encompasses all the notions of confidentiality introduced in previous works on controlled query evaluation. Some assumptions are implicit in the definition: (i) the user may know the algorithm of the controlled evaluation function, (ii) the user is rational, i.e., he derives nothing besides what is implied by his knowledge and the behavior of the database.

2.4 Existing Methods for Known Policies

Table 2 sketches the censors of the enforcement methods for known policies reported previously (i.e., columns 2 and 3 of Table 1) together with the pertinent preconditions in the sense of the new Definition 1. The original definitions are reformulated using the function *pot_sec*(.) (cf. (2)).

Proposition 1 (confidentiality). *The (suitably formalized versions of) previously reported enforcement methods for* known policies *preserve confidentiality in the sense of Definition 1.*

A careful comparison of the different methods summarized in Table 2 shows an interesting relationship between the treatment of potential secrets and secrecies. Each method for secrecies is equivalent to the corresponding method for potential secrets applied to the policy *pot_sec*(*secr*) (where *secr* is the original policy). This is not immediately obvious for refusal, but it can be proved easily.

Proposition 2 (correspondence). *Each (suitably formalized versions of a) previously reported enforcement method for* known secrecies *using a policy instance secr is equivalent to the corresponding method for* known potential secrets *using the policy instance pot_sec*(*secr*).

2.5 Existing Methods for Unknown Policies

For unknown policies, only methods based on secrecies have been introduced so far. The existing methods for unknown secrecies (i.e., column 4 of Table 1) are summarized in the rightmost column of Table 3. Extending Proposition 1, we note that they also satisfy the generalized notion of confidentiality (Definition 1). The left column of Table 3 provides equivalent reformulations of the corresponding methods for known secrecies.

A crucial difference between a method for *known* policies and the corresponding method for *unknown* policies is the following: in the former case the pertinent censor takes care about *all* sentences in *pot_sec* or *pot_sec*(*secr*), respectively, whereas in the latter case only sentences that are *true in the current instance* of the information system are considered. This property makes the censors *instance dependent*, whereas the censors for known policies depend only on the query, the log and the policy.

Table 2. Correspondences between enforcement methods for *known* policies: for each method we indicate the pertinent censor as given in previous work and the precondition as requested by Definition 1. For combined methods, we make the simplifying assumption that $policy \neq \emptyset$ in order to avoid explicit consistency checks for the log

potential secrets $pot\_sec = \{\Psi_1, \ldots, \Psi_k\}$	**secrecies** $secr = \{\{\Psi_1, \neg\Psi_1\}, \ldots, \{\Psi_k, \neg\Psi_k\}\}$ $pot\_sec(secr) = \{\Psi_1, \neg\Psi_1, \ldots, \Psi_k, \neg\Psi_k\}$
lying [4], [2, section 4]: $censor^{\mathrm{ps,known},L}$: $log \cup \{eval^*(\Phi)(db)\} \models pot\_sec\_disj$, where $pot\_sec\_disj := \bigvee_{\Psi \in pot\_sec} \Psi$; $precondition^{\mathrm{ps,known},L}$: $log_0 \not\models pot\_sec\_disj$	**lying** [1, section 5]: $censor^{\mathrm{sec,known},L}$: "not applicable" $precondition^{\mathrm{sec,known},L}$: $log_0 \not\models \bigvee_{\Psi \in pot\_sec(secr)} \Psi$
refusal [2, section 3]: $censor^{\mathrm{ps,known},R}$: (exists $\Psi \in pot\_sec$) $[\, log \cup \{eval^*(\Phi)(db)\} \models \Psi$ or $log \cup \{\neg eval^*(\Phi)(db)\} \models \Psi\,]$; $precondition^{\mathrm{ps,known},R}$: db `model_of` log_0 and (for all $\Psi \in pot\_sec$) $[\, log_0 \not\models \Psi\,]$	**refusal** [9], [1, section 4.2]: $censor^{\mathrm{sec,known},R}$: (exists $\Psi \in pot\_sec(secr)$) [db `model_of` Ψ and $[\, log \cup \{eval^*(\Phi)(db)\} \models \Psi$ or $log \cup \{\neg eval^*(\Phi)(db)\} \models \Psi$ or $log \cup \{\neg eval^*(\Phi)(db)\} \models \neg\Psi\,]\,]$; $precondition^{\mathrm{sec,known},R}$: db `model_of` log_0 and (for all $\Psi \in pot\_sec(secr)$)$[\, log_0 \not\models \Psi\,]$
combined [3, section 3]: $refusalcensor^{\mathrm{ps,known},C}$: (exists $\Psi_1 \in pot\_sec$) $[\, log \cup \{eval^*(\Phi)(db)\} \models \Psi_1\,]$ and (exists $\Psi_2 \in pot\_sec$) $[\, log \cup \{\neg eval^*(\Phi)(db)\} \models \Psi_2\,]$; $lyingcensor^{\mathrm{ps,known},C}$: (exists $\Psi_1 \in pot\_sec$) $[\, log \cup \{eval^*(\Phi)(db)\} \models \Psi_1\,]$ and (for all $\Psi_2 \in pot\_sec$) $[\, log \cup \{\neg eval^*(\Phi)(db)\} \not\models \Psi_2\,]$; $precondition^{\mathrm{ps,known},C}$: (for all $\Psi \in pot\_sec$) $[\, log_0 \not\models \Psi\,]$	**combined** [3, section 4]: $refusalcensor^{\mathrm{sec,known},C}$: (exists $\Psi_1 \in pot\_sec(secr)$) $[\, log \cup \{eval^*(\Phi)(db)\} \models \Psi_1\,]$ and (exists $\Psi_2 \in pot\_sec(secr)$) $[\, log \cup \{\neg eval^*(\Phi)(db)\} \models \Psi_2\,]$; $lyingcensor^{\mathrm{sec,known},C}$: (exists $\Psi_1 \in pot\_sec(secr)$) $[\, log \cup \{eval^*(\Phi)(db)\} \models \Psi_1\,]$ and (for all $\Psi_2 \in pot\_sec(secr)$) $[\, log \cup \{\neg eval^*(\Phi)(db)\} \not\models \Psi_2\,]$; $precondition^{\mathrm{sec,known},C}$: (for all $\Psi \in pot\_sec(secr)$) $[\, log_0 \not\models \Psi\,]$

Table 3. Correspondences between enforcement methods depending on the user awareness of the policy instance: For each method we indicate the suitably reformulated pertinent censor and the precondition as requested by Definition 1

known	**unknown**
lying under secrecies [1, section 5]:	**lying under secrecies** [1, section 5]:
"not applicable" $censor^{\mathtt{sec},\mathtt{known},L}$: $log \cup \{eval^*(\Phi)(db)\} \models \bigvee_{\Psi \in pot\_sec(secr)} \Psi$	$censor^{\mathtt{sec},\mathtt{unknown},L}$: $log \cup \{eval^*(\Phi)(db)\} \models \bigvee_{\Psi \in pot\_sec(secr) \text{ and } db\ \mathtt{model\_of}\ \Psi} \Psi$
$precondition^{\mathtt{sec},\mathtt{known},L}$: $log_0 \not\models \bigvee_{\Psi \in pot\_sec(secr)} \Psi$	$precondition^{\mathtt{sec},\mathtt{unknown},L}$: $log_0 \not\models \bigvee_{\Psi \in pot\_sec(secr) \text{ and } db\ \mathtt{model\_of}\ \Psi} \Psi$
refusal under secrecies [9], [1, sect. 4.2]:	**refusal under secrecies** [9], [1, sect. 4.1]:
$censor^{\mathtt{sec},\mathtt{known},R}$: (exists $\Psi \in pot\_sec(secr)$) [$log \cup \{eval^*(\Phi)(db)\} \models \Psi$ or $log \cup \{\neg eval^*(\Phi)(db)\} \models \Psi$] ;	$censor^{\mathtt{sec},\mathtt{unknown},R}$: (exists $\Psi \in pot\_sec(secr)$) [db `model_of` Ψ and $log \cup \{eval^*(\Phi)(db)\} \models \Psi$] ;
$precondition^{\mathtt{sec},\mathtt{known},R}$: db `model_of` log_0 and (for all $\Psi \in pot\_sec(secr)$) [$log_0 \not\models \Psi$]	$precondition^{\mathtt{sec},\mathtt{unknown},R}$: db `model_of` log_0 and (for all $\Psi \in pot\_sec(secr)$) [if db `model_of` Ψ then $log_0 \not\models \Psi$]

3 Filling in the Gaps

In this section, we shall try to fill in the gaps of Table 1 by analogy with the relationships between the existing methods observed in Section 2.4 and Section 2.5. In the next subsection, we shall derive secure controlled query evaluations for unknown potential secrets from their counterparts for unknown secrecies (Table 3), by analogy with the relationships observed in Table 2. In the second subsection, we shall try to adapt to unknown policies the combined methods for known policies, by analogy with the observations on Table 3. Unfortunately, it will turn out that all the obvious attempts in this direction fail to preserve confidentiality.

3.1 Lies and Refusal for Unknown Potential Secrets

In the case of *known* potential secrets, the controlled query evaluation based on *lies* uses a censor that protects the disjunction of *all* potential secrets. In the case of *unknown* potential secrets, we can take a less restrictive disjunction. Now it is sufficient to take care of only those potential secrets that are valid in the current instance of the information system. Note that in this case the censor actually depends on the current instance, in contrast to the case for known policies.

Formally we define the *uniform lying* method as follows. The $censor^{\mathrm{ps,unknown},L}$ is given by

$$log \cup \{eval^*(\Phi)(db)\} \models true\_pot\_sec\_disj \tag{5}$$

with $true\_pot\_sec\_disj := \bigvee_{\Psi \in pot\_sec \text{ and } db \texttt{ model\_of } \Psi} \Psi$, and the precondition $precondition^{\mathrm{ps,unknown},L}$ is given by

$$log_0 \not\models true\_pot\_sec\_disj\,. \tag{6}$$

Theorem 1 (confidentiality). *Uniform lying for unknown potential secrets preserves confidentiality in the sense of Definition 1.*

Next, we introduce a *uniform refusal* method for unknown potential secrets by following the same ideas adopted above. Formally, the method is defined as follows. The $censor^{\mathrm{ps,unknown},R}$ is given by

$$(\text{exists } \Psi \in pot\_sec)[\, db \texttt{ model\_of } \Psi \text{ and } log \cup \{eval^*(\Phi)(db)\} \models \Psi\,], \tag{7}$$

and the $precondition^{\mathrm{ps,unknown},R}$ by

$$db \texttt{ model\_of } log_0 \text{ and } (\text{for all } \Psi \in pot\_sec)[\text{ if } db \texttt{ model\_of } \Psi \text{ then } log_0 \not\models \Psi\,]. \tag{8}$$

Theorem 2 (confidentiality). *Uniform refusal for unknown potential secrets preserves confidentiality in the sense of Definition 1.*

3.2 Combined Methods for Unknown Policies

We try to adapt the combined method for known potential secrets to unknown such policies, by restricting the censor tests to true secrets only. Due to this restriction, we have to explicitly ensure log consistency. The returned answers are:

$ans_i :=$
`if` $[\, log_{i-1} \cup \{eval^*(\Phi_i)(db)\}$ is inconsistent or
$(\text{exists } \Psi_1 \in pot\_sec)[\, db$ `model_of` Ψ_1 and $log_{i-1} \cup \{eval^*(\Phi_i)(db)\} \models \Psi_1]\,]$
`then`
  `if` $[\, log_{i-1} \cup \{\neg eval^*(\Phi_i)(db)\}$ is inconsistent or
  $(\text{exists } \Psi_2 \in pot\_sec)[\, db$ `model_of` Ψ_2 and $log_{i-1} \cup \{\neg eval^*(\Phi_i)(db)\} \models \Psi_2]\,]$
  `then mum`
  `else` $\neg eval^*(\Phi_i)(db)$
`else` $eval^*(\Phi_i)(db)$

Thus the censor for refusals, $refusalcensor^{\mathrm{ps,unknown},C}$, looks like

$$\begin{array}{l}
[\, log \cup \{eval^*(\Phi)(db)\} \text{ is inconsistent or} \\
(\text{exists } \Psi_1 \in pot_sec)[\, db \texttt{ model_of } \Psi_1 \text{ and } log \cup \{eval^*(\Phi)(db)\} \models \Psi_1\,]\,] \\
\text{and} \\
[\, log \cup \{\neg eval^*(\Phi)(db)\} \text{ is inconsistent or} \\
(\text{exists } \Psi_2 \in pot_sec)[\, db \texttt{ model_of } \Psi_2 \text{ and } log \cup \{\neg eval^*(\Phi)(db)\} \models \Psi_2\,]\,];
\end{array} \tag{9}$$

and the censor for lies, $lyingcensor^{\mathtt{ps,unknown},C}$, looks like

$$
\begin{array}{l}
[\, log \cup \{eval^*(\Phi)(db)\} \text{ is inconsistent or} \\
(\text{exists } \Psi_1 \in pot\_sec)[\, db \ \mathtt{model\_of}\ \Psi_1 \text{ and } log \cup \{eval^*(\Phi)(db)\} \models \Psi_1\,]\,] \\
\text{and} \\
[log \cup \{\neg eval^*(\Phi)(db)\} \text{ is consistent and} \\
(\text{for all } \Psi_2 \in pot\_sec)[\text{ if } db \ \mathtt{model\_of}\ \Psi_2 \text{ then } log \cup \{\neg eval^*(\Phi)(db)\} \not\models \Psi_2\,]\,].
\end{array} \tag{10}
$$

Furthermore, the $precondition^{\mathtt{ps,unknown},C}(., log_0)(db, pot\_sec)$ is specified by

$$
\begin{array}{l}
log_0 \text{ is consistent and} \\
(\text{for all } \Psi \in pot\_sec)\,[\text{ if } db \ \mathtt{model\_of}\ \Psi \text{ then } log_0 \not\models \Psi\,].
\end{array} \tag{11}
$$

Unfortunately, the controlled evaluation method for combined lying and refusal for unknown potential secrets do not preserve confidentiality in the sense of Definition 1.

Example 1. Let $Q = \langle p \vee q \rangle$, $log_0 = \emptyset$, $pot\_sec = \{p \vee q, \neg q\}$ and $db = \{p\}$. Then $ans_1 = \mathtt{mum}$. Now consider any pair (db', pot_sec') satisfying the precondition (11) and returning the same answers as (db, pot_sec). By definition, since $ans_1 = \mathtt{mum}$, $p \vee q$ entails a potential secret Ψ such that $db' \ \mathtt{model\_of}\ \Psi$. Since the vocabulary is $\{p, q\}$ and three out of the four possible interpretations satisfy $p \vee q$, there are two possibilities: either $\Psi \equiv p \vee q$ or Ψ is a tautology. The latter case is not possible because it violates the precondition. It follows that for all the allowed pairs (db', pot_sec') that return $ans_1 = \mathtt{mum}$, $db' \ \mathtt{model\_of}\ p \vee q$, and hence conditions (a) and (b) of Definition 1 cannot be simultaneously satisfied. This proves that the above combined method does not preserve confidentiality.

There are at least two obvious alternative definitions for ans_i, where only Ψ_1 or Ψ_2 (respectively) must be satisfied by db. It can be shown that these two alternatives do not preserve confidentiality, either.

In the light of these negative results, we can assess different methods for *unknown* policies w.r.t. the given state of the art. First, for unknown policies—be they secrecies (see [1]) or potential secrets (see Section 3.1)—the censor for uniform refusal is weaker than the censor for uniform lying. Consequently, we can state a "longest honeymoon lemma" in the sense of [3]. Informally speaking, the lemma says that the method based on uniform refusal does never modify a correct answer *before* the corresponding method based on uniform lies. Of course, this result holds for input pairs that are admissible for both methods. In general, the two preconditions are incomparable.

Second, the censors for uniform refusal should not be further weakened: If we allowed $log \models \Psi$, for some Ψ occurring in the policy such that $db \ \mathtt{model\_of}\ \Psi$, then we could not prevent users from getting a view (given by log) that entails an actual secret. Surely we do not want such a situation. Hence uniform refusal cannot be improved unless some refusals are replaced by lies. However, as shown above, the current attempts at designing combined methods for unknown potential secrets fail to preserve confidentiality.

We conclude that in the framework of unknown policies, the methods based on uniform refusals are currently the best choice from the point of view of the longest honeymoon lemma, and that there are difficulties in improving them.

4 A Systematic View of Controlled Query Evaluation

In this section we develop a more general analysis of the relationships between different policy models and different awareness assumptions. The results of this section justify the recurrent patterns observed in the previous sections.

4.1 Known Policies vs. Unknown Policies

We start with a simple and intuitive result, stating that unknown policies are easier to handle than known policies.

Proposition 3. *If control_eval preserves confidentiality w.r.t. known policies, then control_eval preserves confidentiality w.r.t. unknown policies. The converse is not true.*

The converse of the first part of Proposition 3 holds if the censor of *control_eval* is *instance independent*, that is, for all db and db',

$$censor(\Psi, log, db, policy) = censor(\Psi, log, db', policy)\,.$$

Theorem 3. *If control_eval has an instance independent censor and preserves confidentiality w.r.t. unknown policies, then control_eval preserves confidentiality w.r.t. known policies.*

In other words, if the censor is instance independent, then *control_eval* preserves confidentiality w.r.t. known policies iff it preserves confidentiality w.r.t. unknown policies. Intuitively, this means that in order to take really advantage of the extra degrees of freedom permitted by unknown policies (as suggested by Proposition 3), the censor *must be instance dependent.*

On one hand, this result justifies the difference between the existing censors for known policies and those for unknown policies. On the other hand, this result tells us that even if the instance dependent attempts at defining a secure combined method for unknown policies failed (Section 3.2), nonetheless any significant future approach should keep on searching for an instance dependent censor.

4.2 Potential Secrets vs. Secrecies

In this section we study some general conditions under which the transformation of secrecies into potential secrets (function *pot_sec*(.)) yields a secure query evaluation method. First, the transformation is formalized.

Definition 2. *Let* $control\_eval^{\mathtt{ps},a,e}$ *be any controlled query evaluation based on potential secrets. The* naive reduction *of secrecies to* $control\_eval^{\mathtt{ps},a,e}$ *is the function* $naive\_red(control\_eval^{\mathtt{ps},a,e}) :=$

$$\lambda Q, log_0, db, secr.\ control\_eval^{\mathtt{ps},a,e}(Q, log_0)(db, pot\_sec(secr))\,.$$

Let the precondition of the naive reduction be satisfied by $(Q, log_0, db, secr)$ *iff the precondition of* $control\_eval^{\mathtt{ps},a,e}$ *is satisfied by* $(Q, log_0, db, pot\_sec(secr))$.

The main theorem needs the following definition.

Definition 3. *Let* $censor^{\mathtt{ps},a,e}$ *and* $precond^{\mathtt{ps},a,e}$ *denote the censor and the precondition (respectively) of* $control\_eval^{\mathtt{ps},a,e}$. *We say that* $control\_eval^{\mathtt{ps},a,e}$ *is* insensitive to false potential secrets *iff for all* Q, log_0, db, pot_sec, *and for all sets of sentences* S *whose members are all false in* db,

$$censor^{\mathtt{ps},a,e}(Q, log_0, db, pot\_sec) = censor^{\mathtt{ps},a,e}(Q, log_0, db, pot\_sec \cup S)$$
$$precond^{\mathtt{ps},a,e}(Q, log_0, db, pot\_sec) = precond^{\mathtt{ps},a,e}(Q, log_0, db, pot\_sec \cup S)\,.$$

Note that all the methods for unknown policies introduced so far are insensitive to false potential secrets (this property causes instance dependence, discussed in the previous subsection). Now we are ready to state the main theorem of this subsection.

Theorem 4. *naive_red*$(control\_eval^{\mathtt{ps},a,e})$ *preserves confidentiality (w.r.t.* `sec`, *a and e, cf. Definition 1) whenever* $control\_eval^{\mathtt{ps},a,e}$ *satisfies any of the following conditions:*

1. $a =$ `known` *and* $control\_eval^{\mathtt{ps},a,e}$ *preserves confidentiality (w.r.t.* `ps`, *a and e).*
2. $e = L$ *(lies), the logs are always consistent and entail no potential secrets, all answers are true in the absence of secrets, and all pairs with an empty policy are admissible.*
3. $a =$ `unknown`, $control\_eval^{\mathtt{ps},a,e}$ *preserves confidentiality and is insensitive to false potential secrets.*

Proof. Part 1 Let $precond^{\mathtt{ps}}$ denote the precondition of $control\_eval^{\mathtt{ps},a,e}$. Consider arbitrary Q, log_0, db_1 and $secr_1$, satisfying the corresponding precondition of $naive\_red(control\_eval^{\mathtt{ps},a,e})$. Let Q' be any finite prefix of Q, and consider $\Theta = \{\Psi, \neg\Psi\} \in secr_1$.

Then, by the definitions, $(db_1, pot\_sec(secr_1))$ satisfies $precond^{\mathtt{ps}}$ and

$$\Psi^* = eval^*(\Psi)(db_1) \in pot\_sec(secr_1). \tag{12}$$

Since, by assumption, confidentiality is preserved for known potential secrets, there exists db_2 such that the following holds:

– $(db_2, pot\_sec(secr_1))$ satisfies $precond^{\mathtt{ps}}$, and thus also $(db_2, secr_1)$ satisfies $precond^{\mathtt{sec}}$.

- $(db_1, pot\_sec(secr_1))$ and $(db_2, pot\_sec(secr_1))$ produce the same answers, and hence so do $(db_1, secr_1)$ and $(db_2, secr_1)$.
- $eval^*(\Psi^*)(db_2) = \neg\Psi^*$, and thus also

$$eval^*(\Psi)(db_2) = \neg\Psi^*. \tag{13}$$

From (12) and (13) we conclude that

$$\{eval^*(\Psi)(db_1), eval^*(\Psi)(db_2)\} = \{\Psi^*, \neg\Psi^*\} = \{\Psi, \neg\Psi\}.$$

Hence confidentiality is preserved for known secrecies. This concludes Part 1.

Part 2 Given Q, log_0, db_1 and $secr_1$, let $\Psi \in pot\_sec(secr_1)$ and Q' be any finite prefix of Q. By assumption, there exists always an instance db_2 that satisfies the last log of $control\_eval^{\mathtt{ps},a,e}(Q', log_0)(db_1, pot\_sec(secr_1))$ and falsifies Ψ. It can be verified by a straightforward induction that this evaluation sequence equals

$$control\_eval^{\mathtt{ps},a,e}(Q', log_0)(db_2, pot\_sec(\emptyset))$$

(because all answers must be true by assumption and db_2 is a model of the original answers by construction). Moreover, the pair $(db_2, pot\_sec(\emptyset))$ is admissible by assumption. It follows that the naive reduction preserves confidentiality. This completes Part 2.

Part 3 (Sketch) It suffices to show that for all finite sequences of queries Q', for all log_0, for all admissible pairs $(db_1, pot\_sec(secr_1))$ for $control\_eval^{\mathtt{ps},a,e}$, and for all $\Psi \in pot\_sec(secr_1)$, there exists an admissible pair $(db_2, pot\_sec(secr_2))$ satisfying conditions (a) and (b) of Definition 1, with $policy_1 = pot\_sec(secr_1))$, $policy_2 = pot\_sec(secr_2))$, and $p = \mathtt{ps}$.

By assumption, $control\_eval^{\mathtt{ps},a,e}$ preserves confidentiality, therefore there exists an admissible pair (db_2, pot_sec_2) satisfying conditions (a) and (b) of Definition 1. Since $control\_eval^{\mathtt{ps},a,e}$ is insensitive to false potential secrets, we can assume, without loss of generality, that all the sentences in pot_sec_2 are satisfied by db_2.

Now let $secr_2 = \{\{\Psi, \neg\Psi\} \mid \Psi \in pot\_sec_2\}$. By construction, the sentences in $pot\_sec(secr_2) \setminus pot\_sec_2$ are all false in db_2, and hence (by assumption) $control\_eval(Q', log_0)(db_2, pot\_sec_2) = control\_eval(Q', log_0)(db_2, pot\_sec(secr_2))$ and $(db_2, pot\_sec(secr_2))$ is admissible. Then $(db_2, pot\_sec(secr_2))$ is an admissible pair that satisfies (a) and (b) of Definition 1, as required. □

It is interesting to note that all the existing methods fall into the three cases of the above theorem.

From Theorem 3 and Theorem 4.(1) we immediately get the following result.

Corollary 1. *If the censor of* $control\_eval^{\mathtt{ps},a,e}$ *is instance independent and* $control\_eval$ *preserves confidentiality w.r.t.* ps, *a and e, then the naive reduction* $naive\_red(control\_eval^{\mathtt{ps},a,e})$ *preserves confidentiality w.r.t. both* sec,known,*e, and* sec,unknown,*e.*

Remark 1. The above results subsume neither Theorem 1 nor Theorem 2. In fact, in this section we show how methods for potential secrets can be adapted to secrecies (not viceversa), while the gaps of Table 1 concerned only potential secrets.

The naive reduction does not always preserve confidentiality, even if the underlying $control\_eval^{\mathtt{ps},a,e}$ does.

Example 2. Suppose that the vocabulary is $\{p, q\}$, and $control\_eval^{\mathtt{ps},a,e}$ returns always `mum` for all possible combinations of Q, log_0, db and pot_sec, with two exceptions, where $log_0 = \emptyset$, and either

- $pot\_sec = \{p \vee \neg q, \neg(p \vee \neg q)\}$, $db = \{q\}$, or
- $pot\_sec = \{\neg(p \vee q)\}$, $db = \emptyset$.

In these two cases, let $ans_i := eval^*(\Psi_i)(db)$ if (i) $log_{i-1} \cup ans_i$ does not entail any potential secret, and (ii) both $\{q\}$ and $\emptyset$ are models of $log_{i-1} \cup ans_i$. If these conditions are not simultaneously satisfied, let $ans_i :=$ `mum`.

Finally, the only admissible instances (in all cases) are those that satisfy log_0.

It can be verified that this method preserves confidentiality (hint: the two special cases return the same answers for all query sequences; the other cases are obviously indistinguishable). Now, given $Q = \langle \neg p, q \rangle$, the answers returned for the admissible pair $(\{q\}, \{p \vee \neg q, \neg(p \vee \neg q)\})$ are $\langle \neg p, \mathtt{mum} \rangle$. The only other instance that returns these answers is $\emptyset$ under the policy $\{\neg(p \vee q)\}$, which is *not* closed under negation, i.e., it does not correspond to any set of secrecies. Then the naive reduction does not preserve confidentiality w.r.t. $p = \mathtt{sec}$. □

Note that the above example makes use of a controlled evaluation where (i) the logs are always satisfied by the current instance, and (ii) the answers depend on the false potential secret $p \vee \neg q$. Then the critical assumptions in Theorem 4.(2) (uniform lies) and Theorem 4.(3) (independence from false potential secrets) cannot be simply dropped, because this example would then become a counterexample.

5 Summary and Conclusions

The many works on controlled query evaluation have been given a uniform view in Table 2 and Table 3, and the different notions of confidentiality have been unified by Definition 1. Moreover, we have extended the picture of Table 1 by introducing two new controlled evaluation methods based on lies and refusals for unknown potential secrets (Section 3.1).

Surprisingly, all the attempts at designing secure combined methods for unknown policies failed. The attempts were based on *instance dependent* censors, by analogy with the existing methods for unknown policies. The general results of Section 4.1 tell us that this is the right direction if the extra degree of freedom allowed by unknown policies is to be exploited effectively.

We have introduced a so-called *naive reduction* of secrecies to potential secrets. The naive reduction does not always preserve confidentiality, but we proved in Theorem 4 that confidentiality is guaranteed under fairly natural assumptions. All the known controlled evaluation methods fall into the three points of this theorem, so Table 1 could be entirely reconstructed from the methods based on potential secrets (including the new methods). This fact provides a formal justification of the relationships between different methods empirically observed in Section 2. Thus we achieve a deeper and systematic view of the many forms of controlled query evaluation proposed so far. Moreover, our results show how administrators may freely choose the preferred policy model (secrecies or potential secrets) if the adopted evaluation method is based on potential secrets and fits into one of the cases of Theorem 4. Finer-grained confidentiality requirements can be obtained by mixing secrecies and potential secrets. For each pair of complementary sentences the administrator may decide whether both sentences should be protected, or only one of them should be protected, or none of them needs protection.

We are planning to relate our theoretical studies to the existing approaches to confidentiality, (e.g., polyinstantiation and statistical database perturbation), in order to investigate the practical consequences of our theoretical results. Further future work includes availability policies. Thereby, we hope to get a deeper understanding about the minimality of censors (see for instance [4] or [2], Prop. 4) and the various roles of the user log.

Acknowledgements

We are grateful to the anonymous referees for their careful and cooperative comments and suggestions.

References

[1] Biskup, J.: For unknown secrecies refusal is better than lying, Data and Knowledge Engineering, 33 (2000), pp. 1-23. 40, 41, 44, 46, 47, 49
[2] Biskup, J., Bonatti, P. A. : Lying versus refusal for known potential secrets, Data and Knowledge Engineering, 38 (2001), pp. 199-222. 40, 41, 44, 46, 54
[3] Biskup, J., Bonatti, P. A. : Controlled query evaluation for known policies by combining lying and refusal, Proceedings 2nd Int. Symp. on th Foundations of Information and Knowledge Systems, FoIKS 02, Lecture Notes in Computer Science 2284, Springer, Berlin etc., 2002, pp. 49-66. 40, 41, 46, 49
[4] Bonatti, P. A., Kraus, S., Subrahmanian, V. S.: Foundations of secure deductive databases, IEEE Transactions on Knowledge and Data Engineering 7,3 (1995), pp. 406-422. 40, 41, 46, 54
[5] S. Castano, M. Fugini, G. Martella, P. Samarati: Database Security, Addison-Wesley, 1994. 40
[6] D. E. Denning: Cryptography and Data Security, Addison-Wesley, 1982. 40
[7] Lloyd, J. W.: Foundations of Logic Programming, Springer, 1987. 42
[8] Shoenfield, J. R.: Mathematical Logic, Addison-Wesley, Reading etc., 1967. 42

[9] Sicherman, G. L., de Jonge, W., van de Riet, R. P.: Answering queries without revealing secrets, ACM Transactions on Database Systems 8,1 (1983), pp. 41-59. 40, 41, 46, 47