FOR UNKNOWN SECRECIES REFUSAL IS BETTER THAN LYING

Joachim Biskup

Abstract A shared information system is expected to comply with the following potentially conflicting requirements. It should *provide useful answers* to arbitrary queries, while on the other hand it should *preserve certain secrets* according to a security policy. We study and compare two previously suggested approaches to meet these requirements, namely *refusal* of statements and *lying*. The investigation is performed using a highly abstract and general framework, both with respect to the information system and the preservation of secrets. The assessment shows that for unknown secrecies refusal is better than lying. In particular, while preserving the same secrets refusal can provide more useful answers.

Keywords: Lying, query evaluation, refusal, secrecy

1. INTRODUCTION

An information system is commonly employed as a tool for sharing persistent data among various users, for supporting communications among these users, and for deriving answers to queries on the basis of the stored data. In general, however, a specific user has only restricted rights to the services of the information system. In particular, query answering on behalf of a user can be restricted so as not to reveal some parts of the data which are considered to be secrets. Thus a shared information system is expected to comply with the following potentially conflicting requirements. On the one hand it should *provide useful answers* to arbitrary queries, while on the other hand it should *preserve certain secrets* according to a security policy. We study and compare two previously suggested approaches to satisfy these requirements, namely *refusal* of statements and *lying*. The investigation is performed using a highly abstract and general framework, both with respect to the information system and the preservation of secrets.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35508-5_22](https://doi.org/10.1007/978-0-387-35508-5_22)

V. Atluri et al. (eds.), *Research Advances in Database and Information Systems Security*

© IFIP International Federation for Information Processing 2000

The *information system* framework basically consists of the following features. An *instance* of the information system is seen as a structure of an appropriate logic. A *query* is a sentence of that logic. And finally, a query (sentence) is *evaluated* with respect to an instance (structure) by determining whether the instance is a *model* of the query (or, in other words, whether the query is *true* in the instance).

The *preservation of secrets* framework basically consists of the following features. A *secret* might be any sentence for which the instance is a model. A security *policy* states that a given set of secrets is not allowed to be revealed to a specific user, i.e. the user should not be able to know the secrets on the basis of answers to queries. More precisely, for any instance a secret is *preserved*, if for any sequence of queries there exists an essentially different instance with the following properties: First, the system produces the same answers for the two instances, i.e., the two instances are indistinguishable for the user. Second, while the actual instance is a model of the secret, the other instance is a model of the negation of the secret.

Any approach to achieve preservation of secrets must be based on some assumptions about the *user's capabilities*. In any case these capabilities comprise the knowledge about the *system's strategy* to preserve secrets, and the system's *answers to previous queries*. Additionally, before issuing the first query, the user has some *initial belief* about the instance which may be true or not. Thus, at any time the user's *current belief* is assumed to depend both on his initial belief and the system's answers to previous queries.

A first approach [8] to achieve preservation of secrets is based on *refusal* of statements, i.e. on just returning a special value (say *mum*), whenever the system recognizes that otherwise a secret would be challenged. A second approach [1] is based on *lying*. Deviating from [1], in this paper we use a version of lying that *always* requires to return the negation of the right answer if a challenge is detected. This version appears to be more appropriate within our context. We study both approaches in a *unified framework*, in order to assess and to compare their advantages and disadvantages. The framework is set up in the tradition of previous work on information flow control and inference control, as presented for example in [2, 3, 4, 6].

Our investigations focus on the case that the user supposedly does not know the *secrecies*, i.e. the alternatives of a sentence and its negation one of which is the right answer to be kept secret. The *insight* gained in this paper can be summarized by the statement given as the title of the paper:

- For unknown secrecies, refusal is better than lying.
- In particular, while preserving the same secrets refusal can provide more useful answers.

Table 9.1 Controlled query evaluation with initial belief $\mathbf{b} \Rightarrow \mathbf{s1} \vee \mathbf{s2}$.

query	ordinary query result	answer with refusal	answer with lying	comment
$\mathbf{b} \Rightarrow$ $\mathbf{s1} \vee \mathbf{s2}$	$\mathbf{b} \Rightarrow$ $\mathbf{s1} \vee \mathbf{s2}$	$\mathbf{b} \Rightarrow$ $\mathbf{s1} \vee \mathbf{s2}$	$\mathbf{b} \Rightarrow$ $\mathbf{s1} \vee \mathbf{s2}$	the assumed initial belief is confirmed since no secret is challenged
$\mathbf{a}$	$\mathbf{a}$	$\mathbf{a}$	$\mathbf{a}$	no secret is challenged
$\mathbf{b}$	$\mathbf{b}$	$\mathbf{b}$	$\neg \mathbf{b}$	the disjunction of secrets can be revealed under refusal but not under lying
$\neg \mathbf{s2}$	$\mathbf{s2}$	mum	$\neg \mathbf{s2}$	querying an alternative of a secrecy always results in refusal or lying, resp.

Formally, the insight only applies for our general framework. This framework is general in the sense that many reasonable and sufficiently powerful practical systems are expected to be specialized examples. Thus, the insight widely remains valid also for practical situations.

2. A SIMPLE EXAMPLE

This section is intended to serve two purposes. It exemplifies the high level ideas and claims surveyed in the introduction. And it illustrates the formal concepts and results presented in the rest of the paper.

We assume that the underlying logic of our example information system includes the atomic sentences $\mathbf{a}$, $\mathbf{b}$, $\mathbf{s1}$ and $\mathbf{s2}$, which can be used like propositional variables. Suppose the actual *instance* of the information system is given as an Herbrand structure by $db := \{\mathbf{a}, \mathbf{b}, \mathbf{s1}, \mathbf{s2}\}$. Furthermore, suppose that the security policy requires not to reveal whether $\mathbf{s1}$ or $\neg \mathbf{s1}$ holds and whether $\mathbf{s2}$ or $\neg \mathbf{s2}$ holds. Later on this requirement will be formally represented by the set of so-called *secrecies*, i.e. by $secrecies := \{\{\mathbf{s1}, \neg \mathbf{s1}\}, \{\mathbf{s2}, \neg \mathbf{s2}\}\}$. The actual *secrets* are given by the alternatives that are valid in the instance db , i.e. by $secrets := \{\mathbf{s1}, \mathbf{s2}\}$.

Table 9.1 shows a run of controlled query evaluation under both the refusal approach and the lying approach. In this run the user submits the sequence $\mathbf{a}$, $\mathbf{b}$, $\neg \mathbf{s2}$ of *queries* about literals, after having got a confirmation about his initial belief $\mathbf{b} \Rightarrow \mathbf{s1} \vee \mathbf{s2}$. The *initial belief* is chosen such that it is valid with respect to the actual instance and does not violate the security policy. Thus when the initial belief is queried, then the ordinary query result is returned.

The valid literal **a** is not at all related to the secrets, and thus the system can return the ordinary query result. The literal **b** is the premise of the initial belief the conclusion of which, $s1 \vee s2$, is the disjunction of the secrets. For refusal, at this point of time, there is no need for modification since this disjunction has not explicitly declared to be secret. If later on the disjuncts happen to be queried individually then the system can still refuse the ordinary query results. For lying, the system has to proceed more cautiously: once the user believes that the disjunction of the secrets is valid, the system cannot avoid that later on the user can obtain a valid belief on a secret. Hence the system reacts by lying. Finally, the literal $\neg s2$ is not valid, and thus $s2$ is the ordinary query result. This result must be modified under both approaches since it is a secret. Fortunately, the lie about $s2$ is consistent with the previous lie about the literal **b**.

Comparing refusal and lying for the example we can already make some observations. The formal investigations confirm that they also hold in general. First, whenever the lying approach delivers the ordinary query result, so does the refusal approach. Second, sometimes refusal allows to return the ordinary result but lying does not. Third, in general both approaches have to postulate that the user does not know the secrecies. For (our version of) lying, this remark is obvious; for refusal, it can be justified by the observation that seeing the answer *mum*, the user may examine why the system reacts in this way. It will turn out that the refusal approach can be adapted to the case that the user knows the secrecies, at the price of additional refusals.

3. 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. [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 $eval(\Phi) : DS \rightarrow \{true, false\}$ with $eval(\Phi)(db) := db \text{ model_of } \Phi$, where the boolean operator *model_of* is assumed to be appropriately specified for the logic under consideration. We also use an equivalent formalization where either the queried sentence or its negation is returned: $eval^*(\Phi) : DS \rightarrow \{\Phi, \neg\Phi\}$ with $eval^*(\Phi)(db) := \text{if } db \text{ model_of } \Phi \text{ then } \Phi \text{ else } \neg\Phi$.

We always follow the convention that a sequence of two negation symbols is automatically discarded. Our definition trivially implies that for all instances db and for all queries Φ we have $db \text{ model_of } eval^*(\Phi)(db)$.

We also define the semantic relationship $\models$ for *logical implication* in a standard way, namely by $\Phi \models \Psi$:iff for every structure db with $db \text{ model_of } \Phi$ we also have $db \text{ model_of } \Psi$. The complementary relationship is denoted with $\not\models$.

4. CONTROLLED QUERY EVALUATION

The purpose of the *inference control* is to ensure that an ordinary query result is actually shown to the user only for those cases for which this is not considered to be *forbidden* according to some *security policy*. The parameter of the *security policy* is declared as a set *secr* of *secrecies*, each of which consists of a pair of complementary sentences $\{\Psi, \neg\Psi\}$. Then the policy states that for every secrecy in *secr* the user should not be able to determine—whether by explicit answers or by inferences—which alternative of the secrecy is correct (with respect to the current instance). We define $\text{secrecy}(\Psi) := \{\Psi, \neg\Psi\}$. Each set of secrecies *secr* uniquely determines the *secrets* of any particular instance db as $\text{secrets}_{secr,db} := \{eval^*(\Psi)(db) \mid \text{secrecy}(\Psi) \in secr\}$.

In order to enforce any declared policy the system has to make assumptions about the *user's capabilities*. Since, in this scenario, the user is supposed to potentially behave as an adversary, these assumptions should be as conservative as possible, and they cannot be obtained by cooperation with that user. Our general assumptions will be as follows:

- The user knows the system's strategy of controlled query evaluation, and he has some *initial belief* about the instance.
- The user *remembers all answers* to previous queries.
- The user has *unrestricted computational power* to reason.

Whenever the system recognizes that the ordinary answer to a query would challenge a secret, it basically has two options to react:

- It can *refuse* any substantial statement on the query, say by returning the special value *mum*. Refusal has been suggested in [8]. The main results will be reviewed in Section 5..
- It can *lie* in the sense that it returns the negation of the ordinary query result. Lying has been suggested in [1] in a somehow different version that is based on secrets rather than on secrecies. The main results, suitably adopted for our version and framework, will be restated in Section 6..

Having designed any controlled query evaluation, we have to verify that the expected properties of a security model are indeed met. For our investigations the *security model* comprises the following features:

- The user issues *sequences* of queries.
- For any sequence of queries the system should *protect the secrets*.
- The secrets should be protected in the sense that it is *in principal impossible* for the user to gain complete information about the secrets.
- *Complete information* would mean that the given query answers could result from exactly one choice for the secrets (i.e. for any secrecy $\{\Psi, \neg\Psi\}$ the pre-image of any sequence of answers should contain *at least two* essentially different instances which differ for the secrecy, one of which is a model of Ψ , and the other one is a model of $\neg\Psi$).

The *control facilities* basically consist of three components:

- The first component is a *censor*, which decides whether the ordinary query result is allowed to be shown to the user. We study censors which consider the instance *db*, the secrecies *secr*, the user image *user* and the query Φ . Thus, such a censor returns a decision of the form

$$\text{censor}(db, secr, user, \Phi) \in \{\text{true (do not show result)}, \text{false (show result)}\}.$$
- The second component is a *modifier*, denoted by *modify*, which possibly modifies the ordinary query result. For refusal we will use the *refusal modifier* which constantly returns the special value *mum*. And for lying we will use the *lying modifier* which always returns the negated sentence.
- The third component is the *user image*, denoted by *user*, which contains the system's *explicitly* represented assumptions on the user's belief about the instance. The user image is just a set of sentences in the language of the logic. We will allow the user image to be initialized more or less freely, and for each query the returned answer is inserted but only if that answer is a sentence in the logic.

Since the controlled query evaluation has to keep track of both the initial belief, $user^0$, and the previous answers, a sequence of queries $(\Phi^1, \dots, \Phi^n)$ is processed using a memory. Thus, for each $(\Phi^1, \dots, \Phi^n)$ and $user^0$ we consider a function

$$\text{control_eval}((\Phi^1, \dots, \Phi^n), user^0)$$

which is formalized by an inductive definition of the form:

$$\text{control_eval}((\Phi^1, \dots, \Phi^n), \text{user}^0)(\text{db}, \text{secre}) := \\ ((\text{ans}^1, \text{user}^1), \dots, (\text{ans}^n, \text{user}^n))$$

with

$$\text{ans}^i := \text{if } \text{censor}(\text{db}, \text{secre}, \text{user}^{i-1}, \Phi^i) \text{ then } \text{modify}(\text{eval}^*(\Phi^i)(\text{db})) \\ \text{else } \text{eval}^*(\Phi^i)(\text{db}),$$

$$\text{user}^i := \text{if } \text{ans}^i \text{ is a sentence then } \text{user}^{i-1} \cup \{\text{ans}^i\} \text{ else } \text{user}^{i-1}.$$

Any specific approach is determined by the local functions *censor* and *modify*.

5. REFUSAL OF STATEMENTS

Controlled query evaluation with *refusal* treats queries according to the following rules:

- It delivers the ordinary query result if no secret is challenged.
- Otherwise it *refuses* to give a statement by using the refusal modifier.
- It maintains a user image *user* with “true beliefs”.
- Only arguments $(\text{db}, \text{secre})$ with precondition $\text{db_model_of } \text{user}^0$ are “allowed”.

Obviously we can only protect secrets which are not already initially believed. Hence we are interested in ensuring the *invariant for real secrets*:

$\text{user}^i \not\models X$, for all $X \in \text{realsecrets}$, where

$\text{realsecrets} := \{X \mid \text{user}^0 \not\models X \text{ and } \text{db_model_of } X \text{ and } \text{secrecy}(X) \in \text{secre}\}.$

The main results of [8] about controlled evaluation with refusal are restated in the next subsections. They can be basically summarized as follows:

- With regard to a modelled user who does *not* know the secrecies: for preserving secrets under all sequences of queries, it is necessary and sufficient that the censor maintains the invariant for real secrets. This property is just achieved by the *secret censor*.
- With regard to a modelled user who *is aware* of the secrecies: for preserving secrets under all sequences of queries, it is necessary and sufficient that the censor maintains a stronger invariant—one that protects any sentence of the secrecies under arbitrary answers. This property is just achieved by the *secrecy censor*.

Firstly, we define and study the *secret censor*. The secret censor demands that the ordinary query result be modified if the user could use the correct result $eval^*(\Phi)(db)$ to infer a secret Ψ , i.e.

$$censor_{secrets}(db, secr, user, \Phi) := (\text{exists } \Psi) \\ [db \text{ model_of } \Psi \text{ and } secrecy(\Psi) \in secr \text{ and } user \cup \{eval^*(\Phi)(db)\} \models \Psi].$$

Theorem 1 below is based on the assumption, that the user does *not* know the secrecies. Theorem 2 below shows that this assumption is indeed crucial since otherwise the user can possibly take advantage of refusals (which are not explicitly represented in the user image).

Theorem 1 (successful refusal when secrecies are unknown) *Let censor be the secret censor and modify be the refusal modifcator, $(\Phi^1, \dots, \Phi^n)$ be an arbitrary sequence of queries and $user^0$ be an arbitrary initial belief. Then for all “allowed” arguments $(db_1, secr_1)$ and for all real secrets X with $user^0 \not\models X$ and $db_1 \text{ model_of } X$ and $secrecy(X) \in secr_1$, there exists a different “allowed” argument $(db_2, secr_2)$ with the following properties:*

1. [same answers] $control_eval((\Phi^1, \dots, \Phi^n), user^0)(db_1, secr_1) = control_eval((\Phi^1, \dots, \Phi^n), user^0)(db_2, secr_2);$
2. [different secrets] $eval^*(X)(db_1) \neq eval^*(X)(db_2).$

Proof

Basically, this theorem is Theorem 10.3 from [8]. So we only sketch the proof. According to the assumption, the real secret X has the properties $db_1 \text{ model_of } X$ and $user^0 \not\models X$. Since the secret censor maintains the invariant for real secrets, we also have for the final belief that $user_1^n \not\models X$. Then, by the usual definition of $\models$, there exists a structure db_2 such that $db_2 \text{ model_of } user_1^n$ and $db_2 \text{ model_of } \neg X$.

We take db_2 as the new instance, and we define the new secrecies as

$$secr_2 := \{secrecy(eval^*(\Phi^i)(db_2) \wedge user_1^{i-1}) \mid \\ censor_{secrets}(db_1, secr_1, user_1^{i-1}, \Phi^i)\},$$

where “ $\wedge user_1^{i-1}$ ” abbreviates the conjunctive addition of the conjunction of all elements of $user_1^{i-1}$. The new argument $(db_2, secr_2)$ has the properties stated in the theorem.

□

Theorem 2 *Let censor be the secret censor and modify be the refusal modifcator. Then there exists a sequence of queries $(\Phi^1, \dots, \Phi^n)$, an initial belief $user^0$ and a set secr of secrecies with the following properties:*

there exists a sequence of query answers such that its pre-image for the function $control_eval((\Phi^1, \dots, \Phi^n), user^0)$ with respect to instances only—fixing

the secrecyes—contains exactly one element (up to query equivalence of instances).

Proof

Basically, this theorem is implicitly stated in Section 4 of [8].

□

The theorems also show that, at least in general, the user cannot determine initially unknown secrecyes from the answers, since otherwise he could first do so and then use a reasoning like in the (omitted) proof of Theorem 2 to determine also the secrets. But this method would contradict Theorem 1.

Secondly, we define and study the *secrecy censor*. The secrecy censor demands that the ordinary query result be modified if the user could use an arbitrary answer to infer a secret or its negation, i.e.

$$\begin{aligned} \text{censor}_{\text{secrecies}}(db, \text{secre}, \text{user}, \Phi) := \\ (\text{exists} \Psi) [db \text{ model_of } \Psi \text{ and } \text{secrecy } \Psi \in \text{secre} \text{ and} \\ [user \cup \{eval^*(\Phi)(db)\} \models \Psi \text{ or} \\ user \cup \{\neg eval^*(\Phi)(db)\} \models \Psi \text{ or} \\ user \cup \{\neg eval^*(\Phi)(db)\} \models \neg \Psi]]. \end{aligned}$$

Obviously the secrecy censor is stronger than the secret censor. In particular the secrecy censor maintains a strengthened *invariant for secrecyes*:

$user^i \not\models X$, for all $X \in \text{realsecrets}$, and additionally, if $ans^i \neq \text{mum}$ then $user^{i-1} \cup \{\neg ans^i\} \not\models X$, for all X with $\text{secrecy}(X) \in \text{secre}$.

Theorem 3 (successful refusal when secrecyes are known) *Let censor be the secrecy censor and modify be the refusal modifcator, $(\Phi^1, \dots, \Phi^n)$ be an arbitrary sequence of queries and $user^0$ be an arbitrary initial belief. Furthermore let secre be a fixed set of secrecyes. Then for all instances db_1 such that (db_1, secre) is an “allowed” argument and for all real secrets X with $user^0 \not\models X$ and $db_1 \text{ model_of } X$ and $\text{secrecy}(X) \in \text{secre}$ there exists a different instance db_2 such that (db_2, secre) is “allowed” with the following properties:*

1. *[same answers] $\text{control_eval}((\Phi^1, \dots, \Phi^n), user^0)(db_1, \text{secre}) = \text{control_eval}((\Phi^1, \dots, \Phi^n), user^0)(db_2, \text{secre})$;*
2. *[different secrets] $eval^*(X)(db_1) \neq eval^*(X)(db_2)$.*

Proof

Basically, this theorem is Theorem 10.2 from [8] and proved like Theorem 1.

□

6. LYING

Controlled query evaluation with *lying* treats queries according to the following rules:

- It delivers the ordinary query result if no secret is challenged.
- Otherwise it *lies* by using the lying modifier $modify(\Phi) := \neg\Phi$.
- It maintains a user image *user* which consists of a belief.
- The function $control_eval((\Phi^1, \dots, \Phi^n), user^0)$ is considered only for consistent *initial beliefs* $user^0$.

Controlled query evaluation should maintain *consistency of user belief* as an invariant. This requirement cannot always be achieved if for any query a pertinent though possibly lied statement is returned. The following example characterizes the situation. Define $secrets := \{\Psi_1, \dots, \Psi_k\}$ and $user^0 := \{\Psi_1 \vee \dots \vee \Psi_k\}$. If the user issues the sequence of queries $\Psi_1, \dots, \Psi_k$, then the ordinary query results, namely Ψ_i , must be modified, i.e. the lying modifier generates the answers $\neg\Psi_i$. Hence we finally get the user belief $user^k := \{\Psi_1 \vee \dots \vee \Psi_k, \neg\Psi_1, \dots, \neg\Psi_k\}$, which clearly is inconsistent. The final event, that the user belief becomes inconsistent after k answers, is closely related to the directly preceding event, that the still consistent user belief after $k - 1$ answers logically implies the last secret. For consider that user belief $user^{k-1} := \{\Psi_1 \vee \dots \vee \Psi_k, \neg\Psi_1, \dots, \neg\Psi_{k-1}\}$: clearly we have $user^{k-1} \models \Psi_k$. Hence the initial belief and the first $k - 1$ answers (lies) would reveal the last secret Ψ_k .

The example indicates that in general it is necessary to maintain a strengthened *invariant* that not only preserves the consistency of the user belief but also the assertion that the user belief does not imply the *disjunction of all secrets* (which clearly includes the consistency). The more precise investigations below will provide a main result that can be basically summarized as follows:

- With regard to a modelled user who does *not* know the secrecies: for preserving secrets under all sequences of queries, it is necessary and sufficient that the initial belief does not imply the disjunction of all secrets and that the censor maintains the corresponding invariant. This property is just achieved by the *secret-disjunction censor*.

The *secret-disjunction censor* demands that the ordinary query result be modified if the user could use the correct result to infer the *disjunction of all secrets*, i.e., we define

$$censor_{secdisj}(db, secr, user, \Phi) := user \cup \{eval^*(\Phi)(db)\} \models secret_disjunction,$$

$$secret_disjunction := \begin{cases} \Psi_1 \vee \dots \vee \Psi_k & \text{for } secrets := \{\Psi_1, \dots, \Psi_k\}, \\ \diamond & \text{for } secrets := \emptyset. \end{cases}$$

Then, together with the *lying modifier* $modify(\Phi) := \neg\Phi$, by instantiating the generic declaration given in Section 4., we get *controlled query evaluation with lying*. Theorem 4 below first shows that the *necessary* invariant for the disjunction of all secrets is indeed maintained. Theorem 4 and Theorem 5 together say that the invariant is also *sufficient* for preserving the secrets. These theorems are related to the assertions in Section V of [1], which were made there within a model-theoretic framework for a modal logic of belief and for a version of lying based on secrets rather than on secrecyes.

Theorem 4 (Invariant of the secret-disjunction censor) *Let censor be the secret-disjunction censor and modify be the lying modifier. Then controlled query evaluation with lying has the following property:*

$$user^{i-1} \not\models secret_disjunction$$

if and only if

$$user^i \not\models secret_disjunction.$$

Proof

“ $\Leftarrow$ ”: The claim immediately follows from $user^{i-1} \subseteq user^i$.

“ $\Rightarrow$ ”: Assume $user^{i-1} \not\models secret_disjunction$, and let Φ^i be an arbitrary query. According to the definition of the controlled query evaluation with lying, the returned answer ans^i is either $eval^*(\Phi^i)(db)$ or $\neg eval^*(\Phi^i)(db)$, and this answer is added to $user^{i-1}$.

Case 1: $ans^i = eval^*(\Phi^i)(db)$. This case happens iff $user^{i-1} \cup \{eval^*(\Phi)(db)\} \not\models secret_disjunction$.

Case 2: $ans^i = \neg eval^*(\Phi^i)(db)$. This case happens iff $user^{i-1} \cup \{eval^*(\Phi)(db)\} \models secret_disjunction$.

Now suppose indirectly that we also have

$$user^{i-1} \cup \{\neg eval^*(\Phi)(db)\} \models secret_disjunction.$$

Then, by a well-known result of logic, these implications are only possible if even $user^{i-1} \models secret_disjunction$. This relationship, however, contradicts the assumption.

□

Theorem 5 (successful lying when secrecyes are unknown) *Let censor be the secret-disjunction censor and modify be the lying modifier, $(\Phi^1, \dots, \Phi^n)$*

be an arbitrary sequence of queries and $user^0$ be an arbitrary initial belief. Then for all arguments $(db_1, secr_1)$ with $user^0 \not\models secret_disjunction_1$ and for all secrets X with $db_1 \text{ model_of } X$ and $secrecy(X) \in secr_1$, there exists a different argument $(db_2, secr_2)$ with the following properties:

1. [same answers] $control_eval((\Phi^1, \dots, \Phi^n), user^0)(db_1, secr_1) = control_eval((\Phi^1, \dots, \Phi^n), user^0)(db_2, secr_2)$;
2. [different secrets] $eval^*(X)(db_1) \neq eval^*(X)(db_2)$.

Proof

Setting $user_1^0 = user^0$ we have by assumption

$$user_1^0 \not\models secret_disjunction_1$$

and hence, by Theorem 4 about the invariant, also

$$user_1^n \not\models secret_disjunction_1.$$

Since $X \in secrets_1$ and thus $X \models secret_disjunction_1$, we obtain $user_1^n \not\models X$. Then, by the usual definition of $\models$, there exists a structure db_2 such that

$$db_2 \text{ model_of } user_1^n \text{ and } db_2 \text{ model_of } \neg X.$$

We take db_2 as the new instance, and we define the new secrecies as $secr_2 := \emptyset$. Hence we have $secret_disjunction_2 := \diamond$. Then the new argument $(db_2, secr_2)$ has the properties stated in the theorem. □

7. ASSESSMENT OF REFUSAL AND LYING

Our assessment applies for *uniform* controls to defend against *unrestricted* users, as explained below. First, the controls should preserve the secrets *uniformly*, i.e. for all possible sequences of queries and user images, and for all “allowed” arguments the controls should be performed with the same security mechanisms (the censor and the modifier in particular). Preserving secrets means, here, that in *all* cases the user cannot uniquely determine the secret part of the current instance. Second, this property holds even if the user can employ *unrestricted* computational power. For, whatever sequence of queries is issued and whatever initial belief is available, the evaluation function turns out to be “nowhere injective with respect to the secrets”, as far as it is successful.

Then, in Table 9.2 we summarize the properties of the approaches to controlled query evaluation as treated in this paper.

Refusal using the secret censor has the following *advantages* in comparison to (our version of) lying using the secret-disjunction censor:

- Refusal is applicable for a *larger class* of normal arguments.

Table 9.2 Assessment of uniform control mechanisms to defend against unrestricted users.

	<i>Refusal with secret censor</i>	<i>Refusal with secrecy censor</i>	<i>Lying with secret-disjunction censor</i>
<i>censor</i>	<p>censorship, if current user belief and (correct) ordinary query result imply a secret</p>	<p>censorship, if current user belief and arbitrary result imply a secret or its negation</p>	<p>censorship, if current user belief and (correct) ordinary query result imply the disjunction of all secrets</p>
<i>modifier</i>	<p>refusal modifier: $modify() = \text{mum}$</p>	<p>refusal modifier: $modify() = \text{mum}$</p>	<p>lying modifier: $modify(\Phi) = \neg\Phi$</p>
<i>user image</i>	<p>user belief $user$ with $dbmodel_of\ user$</p>	<p>user belief $user$ with $dbmodel_of\ user$</p>	<p>user belief $user$ with $user$ consistent</p>
<i>precondition</i>	<p>argument allowed: $dbmodel_of\ user^0$</p>	<p>argument allowed: $dbmodel_of\ user^0$</p>	<p>initial user belief $user^0$ does not imply the disjunction of all secrets</p>
<i>exceptional case</i>	<p>initial user belief $user^0$ implies a secret, and thus all statements are refused: mechanism not applicable</p>	<p>initial user belief $user^0$ implies a secret, and thus all statements are refused: mechanism not applicable</p>	<p>initial user belief $user^0$ implies the disjunction of all secrets (in particular the initial belief $user^0$ is inconsistent): mechanism not applicable</p>
<i>normal case</i>	<p>initial user belief $user^0$ does not imply any secret</p>	<p>initial user belief $user^0$ does not imply any secret</p>	<p>initial user belief $user^0$ does not imply the disjunction of all secrets (and thus is not inconsistent)</p>
<i>invariant in normal case</i>	<p>user belief $user$ does not imply any secret</p>	<p>user belief $user$ does not imply any secret or its negation, even if the last non-refused ordinary query result is substituted by its negation</p>	<p>user belief $user$ does not imply the disjunction of all secrets</p>
<i>secrecies known</i>	<p>secrets cannot be preserved: Theorem 2</p>	<p>secrets can be preserved: Theorem 3</p>	<p>secrets cannot be preserved: obvious</p>
<i>secrecies unknown</i>	<p>secrets can be preserved: Theorem 1</p>	<p>secrets can be preserved: Theorem 3</p>	<p>secrets can be preserved: Theorem 5</p>

- The secret censor is weaker than the secret-disjunction censor, and thus the former *more often* shows the ordinary query result than the latter.
- By refusal, the user is *never misled* by lying.
- Refusal can be *combined* with the secrecy censor, in order to cope with applications where the secrecies are supposedly known to the user.

However, there also could be two *disadvantages*:

- The precondition of refusal is stronger than the precondition of lying. The former requires that the assumed initial belief is *compatible with the actual instance*, whereas the latter only demands its *consistency*.
- Refusal reveals the pure fact of modification (but neither the specific reasons behind the refusal nor, in particular, the secrecies) whereas, of course, lying is not indicated.

In conclusion, for the majority of applications we expect that the advantages of refusal will be ranked higher than its disadvantages. Therefore, in general we favour refusal over lying.

There are several directions for further interesting research. First, we could relax the two features emphasized above. Second, we could investigate how to combine refusal and lying and possibly even other types of reactions, in particular for nonuniform mechanisms to defend against restricted users. Third, we could extend the comparison for the original version of lying that assumes that the user knows that a fixed set of secrets (not secrecies) are to be kept secret. We conjecture that the original version is closely related to refusal with the secrecy censor. Fourth, we could study appropriate instantiations of modal logic for reasoning about the belief of users from the point of view of the protecting system. Fifth, we could refine the investigation for more practical versions of information systems. The previous work on lying [1] and the study in [3] are examples of useful starting points for the last two suggestions. Finally, it would be worthwhile to embed the present and future work into the general theory of Reasoning about Knowledge as elaborated in [5].

Acknowledgments

I sincerely thank Piero Bonatti, David Embley, Ralf Menzel, Barbara Sprick, Reind van de Riet and three anonymous reviewers for helpful comments on previous versions of this paper.

References

- [1] Bonatti, P.A., Kraus, S., Subrahmanian, V.S. (1995). Foundations of secure deductive databases. *IEEE Transactions on Knowledge and Data Engineering* 7(3), pp. 406–422.
- [2] Cohen, E. (1977). Information transmission in computational systems. *Proceedings of the Sixth Symposium on Operating Systems Principles*, pp. 133–139.
- [3] Cuppens, F., and Trouessin, G. (1994). Information flow controls vs. inference controls: an integrated approach. *Proceedings of the Third European Symposium on Research in Computer Security*, ESORICS 94. Lecture Notes in Computer Science 875, pp. 447–468. Berlin etc.: Springer.
- [4] Denning, D.E. (1982). *Cryptography and Data Security*. Reading etc.: Addison-Wesley.
- [5] Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y. (1995). *Reasoning about Knowledge*. Cambridge-London: MIT Press.
- [6] Goquen, J.A., and Mesequer, J. (1984). Unwinding and inference control. *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 75–86.
- [7] Shoenfield, J.R. (1967). *Mathematical Logic*. Reading etc.: Addison-Wesley.
- [8] Sicherman, G.L., de Jonge, W., van de Riet, R.P. (1983). Answering queries without revealing secrets. *ACM Transactions on Database Systems*, 8(1), pp. 41–59.