Selected Results and Related Issues of Confidentiality-Preserving Controlled Interaction Execution

Joachim Biskup(✉)

Technische Universität Dortmund, Dortmund, Germany
joachim.biskup@cs.tu-dortmund.de

Abstract. Controlled Interaction Execution has been developed as a security server for inference control shielding an isolated, logic-oriented information system when interacting over the time with a client by means of messages, in particular for query and transaction processing. The control aims at preserving confidentiality in a formalized sense, intuitively and simplifying rephrased as follows: Even when having (assumed) a priori knowledge, recording the interaction history, being aware of the details of the control mechanism, and unrestrictedly rationally reasoning, the client should never be able to infer the validity of any sentence declared as a potential secret in the security server's confidentiality policy. To enforce this goal, for each of a rich variety of specific situations a dedicated censor has been designed. As far as needed, a censor distorts a functionally expected reaction message such that suitably weakened or even believably incorrect information is communicated to the client. In this article, we consider selected results of recent and ongoing work and discuss several issues for further research and development. The topics covered range from the impact of the underlying logic, whether propositional or first-order or for non-monotonic beliefs or an abstraction from any specific one, to the kind of the interactions, whether only queries or also view publishing or updates or revisions or even procedural programs.

Keywords: A priori knowledge · Belief · Censor · Client state · Completeness · Confidentiality · Constraint satisfaction · Distortion · Evaluated secrecy · First-order logic · Guarded commands · Inference control · Information system · Information flow control · Interaction history · Knowledge · Lying · Model theory · Monitoring · Non-monotonic reasoning · Policy · Possibilistic secrecy · Proof theory · Program execution · Query answering · Rational reasoning · Refusal · Relational database · Security automaton · Security invariant · Theorem proving · Update processing · View publishing · Weakening

1 Introduction

As surveyed in [11,12,17,34], Controlled Interaction Execution, CIE, has been developed as a *security server* for *inference control* [9,54] shielding an isolated

M. Gyssens and G. Simari (Eds.): FoIKS 2016, LNCS 9616, pp. 211–234, 2016.
DOI: 10.1007/978-3-319-30024-5_12

information system when *interacting* over the time with a *client* by means of *messages*. Controlled interactions might comprise *query* answering, *update* processing complemented with refreshment notifications, *revision* processing, more generally *transaction* processing, even more generally execution of a *procedural program* with guarded commands, and *view publishing*, in each case based on logic-based *formal semantics* [1,63], like for relational databases.

Following the spirit of many other works on secrecy [28,61], a CIE-control aims at provably *preserving confidentiality* in a fully formalized sense, intuitively and simplifying rephrased as follows: Even when having (assumed) *a priori knowledge*, recording the *interaction history*, being aware of the *details of the control mechanism*, and unrestrictedly *rationally reasoning*, the client should never be able to infer the validity of any sentence declared as a potential secret in the server's *confidentiality policy*. In other words, the client should always believe in the *possibility* that such a sentence is not valid in the underlying information system, or at least not plausible. If interactions may modify the instance of the information system, this requirement refers to either the current instance only or to previous instances as well. Moreover, the *notion of validity* might depend on the kind of the underlying information system, e.g., whether seen as providing a formal and either complete or incomplete representation of an outside "real world", or whether treated as formally reflecting somebody's internal *belief* under non-monotonic reasoning.

To enforce this goal, for each of a rich variety of specific situations a dedicated *censor* has been designed. Basically, on a client's request or triggered by a spontaneous activity of the information system, such a censor first inspects the functionally expected interaction behavior, whether it would preserve confidentiality in the strong sense sketched above. If it does, the *expected* reaction message is sent to the client. Otherwise, the censor determines a *distorted* reaction message that first of all preserves confidentiality and additionally should be as informative as possible for the sake of the conflicting goal of availability. Distortions will lead the system to communicate suitably *weakened* or even believably *incorrect* data to the client, depending on the basic enforcement strategy of the chosen censor. In particular, the choice has to consider whether reaction messages containing lies are seen as socially acceptable for the concrete application.

In principle, at any point of time, the decision taken and the distortions made by the censor not only have to consider the *past* of the interactions but also have to ensure the option to continue with interactions in the *future*. Accordingly, the effectiveness of each censor is based on maintaining a suitably formed *security invariant*. In a sense, the control instantiated by a censor is proceeding like a *security automaton* [7,51,74], which *monitors* an unlimited *stream* of messages built from a client's requests and the corresponding reactions.

Typically, checking the pertinent invariant for a tentative reaction message requires to solve one or several *entailment problems* in the formal logics on which the underlying information system is based or by which the client's reasoning is assumed to be captured, respectively. Hence, from an algorithmic point of view, in general the censor has to be supported by applicable *theorem provers*.

For special cases, however, we would prefer to exploit more dedicated procedures to enhance the runtime efficiency.

In this article, we consider a selection of the results of recent and ongoing work about CIE and discuss several issues for further research and development:

- In Sect. 2, within a simple framework based on finite classical propositional logic, we introduce into the main concepts of CIE for controlling sequences of queries and in particular present the basic approaches to construct censors employing refusals as the strongest form of weakening, lying, and a combination of refusal and lying, respectively.
- In Sect. 3 we abstract from using a specific logic, in particular to compare the basic approaches and to determine their inherent complexity.
- In Sect. 4 we examine the problems arising from essentially increasing the expressiveness of the underlying logic, more specifically of using first-order logic as a foundation of relational databases, in particular enabling to deal with open queries with the need to control completeness sentences.
- In Sect. 5 we describe a static alternative to dynamic query processing, namely to publish a controlled view, basically expanding on two fundamental strategies, an intensionally working one based on sufficiently exhaustive querying and an extensionally working one based on removing violations of constraints stemming from a priori knowledge and the confidentiality policy.
- In Sect. 6 we examine the impact of a more advanced information system, in particular handling incompleteness and belief rather than complete knowledge and the client's corresponding possibilities of inferences.
- In Sect. 7 we extend the interactions to also process updates or revisions, and even to execute a procedural program, finally leading to combine CIE with language based information flow control with declassification.

When considering formal theorems presented in previous work, we will often neglect technical details and omit precise suppositions in order to focus on the main assertion in hopefully intuitive terms. Accordingly, we will refer to such a rephrasement as a "Result", and the reader is kindly advised to find the missing technicalities in the original publications. Moreover, we do not repeat technically elaborated examples. Furthermore, we will summarize an outlook to future work as an "Issue", also mostly in simplifying terms and leaving open the exact status.

As a guideline for reading the remainder, the *overall conclusion* will be the following. A computing agent's reasoning about its *own* knowledge or belief has been a successfully explored research topic, the results of which are first of all used for the information system agent in our context. This research has also been extended to the considerations of one agent, in our context the client, about the *internal* knowledge or belief of *another* agent, in our context the information system, based on *observable* communication data. Now, the goal of inference control adds a further challenge: How can the latter agent, the information system, *minimally distort* communication data in order to confine the achievements in reasoning by the former agent, the client, according to a declarative *confidentiality policy* to be enforced by the information system's security server?

2 A Simple Propositional Framework

We start our considerations with a simple *logic-oriented* information system: A *query* is expressed as a sentence of the language $\mathcal{L}_{pl}$ of classical propositional logic over a finite set of propositional atoms, and an *instance* of the information system is just a (semantic) model represented by a complete truth assignment to the atoms. The information system stores a fixed instance *db* and then would grant the right to submit queries to the client without, however, permitting any direct access to the instance. Moreover, initially the client is assumed to only know a priori that a set *prior* of sentences is satisfied by (valid under) *db*.

Further on, at each point in time i, the client submits a query request with a discretionarily specified sentence φ_i, and – without any control – the information system would then return either φ_i or $\neg\varphi_i$, depending on the truth evaluation $eval(db, \varphi_i)$ of the i-th query sentence regarding the fixed instance, i.e., whether or not $db \models \varphi_i$. Accordingly, after the i-th interaction, the client would be able to infer that *db* satisfies the elements of the current *"syntactic" view* $synView_i := prior \cup \{eval(db, \varphi_1), \ldots, eval(db, \varphi_i)\}$, together with all sentences entailed by that set. For any other sentence ψ, from the point of view of the client, it would appear to be possible that ψ is not satisfied.

Thus, the closure of that set under entailment, in this context treated as the current *"semantic" view* denoted by $semView_i$, would constitute the client's current knowledge about the stored instance. Clearly, without control, the client could obtain complete knowledge about the instance, just by submitting a suitable sequence of queries. Accordingly, the (owner of the) information system would potentially share all information about the instance with the (human user of the) client.

Though *sharing* information would be the main goal of permitting the client to submit any sequence of queries, and thus in effect to learn their actual truth evaluations, the information system's owner might nevertheless want to enforce some *exceptions* for certain sentences seen as being too sensitive and in this context referred to as *potential secrets*. For such a sentence, independently of the actual truth value, from the point of view of the client it should *always* appear to be possible that the sentence is *not* satisfied. This goal could be achieved in two steps. In a first *declarative* step, the owner defines a *confidentiality policy psec* containing all sentences to be treated as a potential secret.

In a second *enforcing* step, the original information system is shielded by a security server for inference control by a censor, which gets the policy as an input parameter. The control then *intercepts* each query request and, only as far as needed, the censor *distorts* the correct truth evaluation $eval(db, \varphi_i)$ of the query sentence into a controlled answer sentence ans_i, in order to confine the information content of the reaction message returned to the client appropriately, as required by the policy. Consequently, the syntactic material available to the client becomes $synView_i := prior \cup \{ans_1, \ldots, ans_i\}$.

Now it is important to observe that the distortions might have broken the straightforward relationship between a syntactic view, literally extracted from the messages of the interactions, and the corresponding semantic view:

- Purely functionally, without control and according to common usage of formal logic, the semantic view is obtained by applying the closure of the syntactic view under entailment.
- With inference control, however, facing potential distortions, the semantic view can only be determined by considering the details of the censor.

More specifically, in the case of inference control, the client has to investigate questions of the following kind. Why did the censor require to return the "verbatim" answer ans_i to the query about the truth evaluation of φ_i? Which possible instances of the information system do lead to that verbatim answer? Which of the two possible truth evaluations of φ_i do cause that verbatim answer? Thus, in most general mathematical terms (see, e.g., Sect. 4 of [9] for further exposition), the semantic view has to be determined by inverting the function that describes the censor on the function values observed as verbatim answers. If the *inverted function* happens to map a verbatim answer to a singleton pre-image containing *exactly one* element, and the client can actually compute this element, then this element contributes full knowledge to the semantic view of the client; the distortions might have changed the syntactic form of the correct answer, but the "real" information content has been preserved. Otherwise, if the pre-image has *at least two* elements, the distortions have not only changed the syntactic form of correct answers but also introduced *uncertainty* about them.

Having the distinction between a syntactic view and the corresponding semantic view in mind, one can construct a concrete censor following three guidelines:

1. Let the censor express any answer, whether correct or distorted, *as a sentence* of the underlying language $\mathcal{L}_{pl}$ (or as a convenient abbreviation of such a sentence) such that the answer looks like "being informative" and the syntactic view $synView_i$ remains a consistent subset of $\mathcal{L}_{pl}$.
2. Let the censor maintain a suitable *security invariant*, also to be ensured as a *precondition* for $synView_0 := prior$, which in particular expresses that none of the potential secrets in the policy *psec* is ever entailed by the *syntactic* view $synView_i$:
$$\text{for all } \psi \in psec : synView_i \not\models \psi\,. \tag{1}$$
Since for propositional logic the semantic notion of *entailment*, $\models$, is equivalent to a syntactic notion of *derivability* (*formal provability*), $\vdash$, given a tentative answer the censor can *computationally* check whether the invariant would be maintained.
3. For each query, in general also dependent on the history and thus on the current view, let the censor *computationally* check this *derivation problem* expressed in the logic – and possibly further or more general ones –, to determine the need of a distortion regarding the *semantic* view (for which the inverted censor function is involved). Then, as indicated by the outcomes of the checks, let the censor form the answer sentence such that, from the client's point of view, it remains *indistinguishable* what the correct answer would have been, i.e., the inversion of the answer would show a pre-image containing both possibilities.

Obviously, the third guideline is the most difficult one to handle, since it is directed to capture the crucial relationship between the syntactic view (what has been shown to the client) and the semantic view (what was the cause of what has been shown). The basic approaches to the wanted construction successfully handle this difficulty by proceeding as sketched in the following.

A censor following the basic *refusal* approach [8,10,13,14,17,36,76] first checks whether the correct answer could already be inferred from the current view; if this is not the case, then – in particular to ensure indistinguishability by instance-independence – the censor inspects both the query sentence φ_i and its negation $\neg\varphi_i$: if returning any of them would lead to a direct violation of the confidentiality policy, then the answer sentence is formed by *weakening* the correct answer into a tautology expressing "tertium non datur" for the query sentence (which w.l.o.g. can be abbreviated by a keyword like `mum`, interpreted as a refusal notification):

$$
\begin{array}{l}
ans_i := \\
\texttt{if } synView_{i-1} \models eval(db,\varphi_i) \\
\texttt{then } eval(db,\varphi_i) \qquad\qquad \%\text{the correct answer} \\
\texttt{else if } (\text{exists } \psi)[\psi \in psec \text{ and} \\
\quad (synView_{i-1} \cup \{\varphi_i\} \models \psi \text{ or } synView_{i-1} \cup \{\neg\varphi_i\} \models \psi)] \\
\quad \texttt{then } (eval(db,\varphi_i) \vee \neg eval(db,\varphi_i)) \% \text{ a tautology, or } \texttt{mum} \\
\quad \texttt{else } eval(db,\varphi_i) \qquad\qquad \% \text{ the correct answer}
\end{array}
\tag{2}
$$

A censor following the basic *lying* approach [8,13,14,17,36,39] only inspects the correct truth evaluation $eval(db,\varphi_i)$ of the query sentence φ_i but – in particular to ensure consistent answers – regarding a stronger violation condition, namely whether the disjunction of all policy elements would be entailed:

$$
\begin{array}{l}
ans_i := \\
\texttt{if } synView_{i-1} \cup \{eval(db,\varphi_i)\} \models \bigvee_{\psi \in psec} \psi \\
\texttt{then } \neg eval(db,\varphi_i) \text{ \% a lie} \\
\texttt{else } eval(db,\varphi_i) \quad \% \text{ the correct answer}
\end{array}
\tag{3}
$$

A censor following the basic *combined* approach [14,15,36] first inspects the correct truth evaluation $eval(db,\varphi_i)$ of the query sentence φ_i; if it would lead to a direct violation then – in particular to ensure consistent answers – the censor additionally inspects the negation of the correct truth evaluation: if that negation would also lead to a violation, then the answer sentence is formed by *weakening* the correct answer into a tautology (or `mum`); otherwise the negation is returned as a lie:

$$
\begin{array}{l}
ans_i := \\
\texttt{if } (\text{exists } \psi)[\psi \in psec \text{ and } synView_{i-1} \cup \{eval(db,\varphi_i)\} \models \psi] \\
\texttt{then if } (\text{exists } \psi)[\psi \in psec \text{ and } synView_{i-1} \cup \{\neg eval(db,\varphi_i)\} \models \psi] \\
\quad \texttt{then } (eval(db,\varphi_i) \vee \neg eval(db,\varphi_i)) \text{ \% a tautology, or } \texttt{mum} \\
\quad \texttt{else } \neg eval(db,\varphi_i) \qquad\qquad \% \text{ a lie} \\
\texttt{else } eval(db,\varphi_i) \qquad\qquad \% \text{ the correct answer}
\end{array}
\tag{4}
$$

Result 1 (Effectiveness of Basic Censors for Query Sequences). *For the propositional framework (and any similar ones) used for controlling* sequences of queries*, each of the basic censors for* refusal*,* lying*, or the* combination *of refusal and lying, respectively,* preserves confidentiality*, i.e.,*

> *for each actual instance, for each confidentiality policy, for each potential secret in that policy, for each assumed a priori knowledge, and for each sequence of query sentences, there exists an alternative instance that satisfies the a priori knowledge as well, generates the same controlled answer sentences, but does not satisfy the potential secret.*

The proofs are based on a structurally quite simple argument outlined as follows. We consider any potential secret $\psi \in psec$. First, at each point in time i, the applicable security invariant ensures the existence of an "alternative instance" that satisfies the current syntactic view but not ψ. Second, a more or less sophisticated induction up to i shows that the actual instance and the alternative instance generate the same controlled answers, and thus are indistinguishable from the client's point of view. Hence, the "alternative instance" is a witness for the *possibility* that the potential secret ψ is *not* valid.

Similarly, as already observed above, a client could gain some kind of best achievable knowledge about the actual instance by submitting an *exhaustive sequence of queries* consisting of all possible queries (up to equivalences). Clearly, the security server can use the same approach for controlled *view publishing*: on request or discretionarily, the censor just generates the final (syntactic) view as the limit of the intermediate views and then sends it to the client. So we have the following corollary to the preceding result.

Result 2 (Effectiveness of Basic Censors for Published Views). *For the propositional framework (and any similar ones) used for controlled* view publishing*, for each of the basic censors for* refusal*,* lying*, or the* combination *of refusal and lying, respectively, the limit of the controlled answers of any exhaustive query sequence* preserves confidentiality*.*

The simple framework suggests several dimensions of elaborating more sophisticated and more comprehensive approaches. In fact, many works on CIE have been motivated this way. In the remainder, we will review and discuss some of these dimensions, as announced in the introduction. Besides considering any of these dimensions in isolation, it would be worthwhile to explore which instantiations of the dimensions are compatible, or could be smoothly composed by suitable constructions.

Issue 1 (Compositionality). *Identify* composition *guidelines for suitably combining features of different dimensions, and establish the corresponding formal* assurances *regarding preservation of confidentiality.*

To actually design and implement a control mechanism as sketched so far, we could employ an architecture as roughly visualized by Fig. 1, which is built from at least the following components:

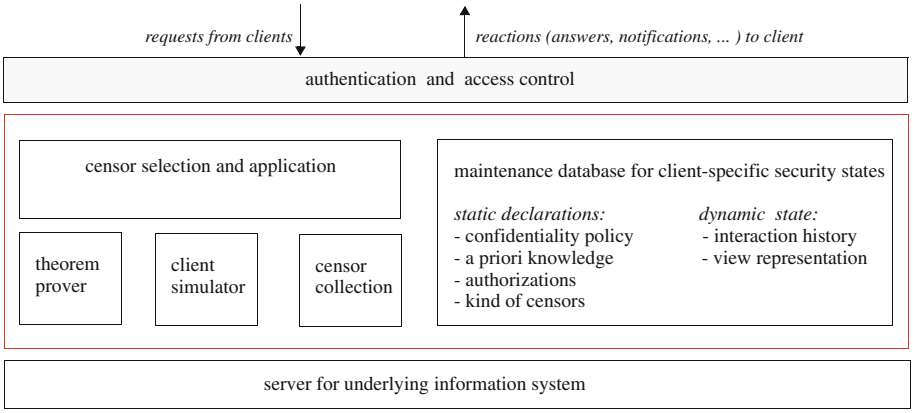

Fig. 1. Rough architecture of Controlled Interaction Execution

- a *functional server* for the underlying information system for storing the instance and (correct) interaction processing;
- a *theorem prover* for solving entailment problems for the logics involved;
- a collection of *censors*, each of which has been verified to meet the confidentiality requirement;
- a *maintenance* database which stores for each authorized client a *client state*, in particular comprising *statically* declared parameters including
  1. the wanted client-specific *confidentiality policy*,
  2. the assumed *a priori knowledge*,
  3. the *authorizations* for interactions, and
  4. the kind of *censors* that could be applied,

  as well as *dynamic* information about
  5. the *interaction history* including the actually applied censors and
  6. a *view representation* of the client's views according to previously returned messages (in the simplest case just *synView*);
- a client *simulator* that determines that representation (in the simplest case just by adding the answer sentences to a log file, which has been initialized with the a priori knowledge).

Issue 2 (Comprehensive System Architecture). *Refine the roughly sketched architecture to an extendible software package which– given suitable parameters for each of the dimensions – can uniformly be configured and then employed as a comprehensive implementation of CIE, and of related and compatible security techniques as well.*

3 An Abstract Framework

In any logic-oriented framework, whether a simple propositional one as sketched in Sect. 2 or a suitably extended one, two aspects are combined:

– an underlying classical (or even non-classical) *logic* comprising an intuitively expressive syntax and a formal notion of either entailment based on models or, if applicable even equivalently, of computational derivability, and
– a *censor function* together with its inverted function, where we deal with reasoning about employing that or a somehow related logic.

Since in general the latter aspect appears to be not directly expressible in the respective logic, one could attempt to bridge the gap between the two aspects by dealing with both of them in a purely functional manner.

Such a unifying treatment could be useful for several purposes, in particular for identifying features that are common to several logics and for separating the computational complexity stemming from the underlying logic and the computational complexity of inference control as essentially encoded in the inverted censor function. These and further considerations have motivated the abstract framework presented in [17]. This framework is inspired by the model-theoretic approach to semantics of formal logics, but without dealing with any concrete syntax.

More specifically, an *information system* is thought to be given by the set $\mathcal{I}$ of its possible abstract *instances* (or data sources), which are functionally treated like (semantic) models in a logic. An *abstract query* φ is then identified with its meaning, namely with a subset of $\mathcal{I}$, such that the evaluation $eval(db, \varphi)$ for an instance db just checks whether or not $db \in \varphi$ (which corresponds to $db \models \varphi$ in a logic framework) and then returns either $\varphi \subseteq \mathcal{I}$ or $(\mathcal{I} \setminus \varphi) \subseteq \mathcal{I}$. Accordingly, if an (abstract) user wants to learn about the *conjunction* of two queries φ_1 and φ_2 and submits them accordingly, he would get $\varphi_1 \cap \varphi_2$ and, similarly, $\varphi_1 \cup \varphi_2$ for the *disjunction*. Thus, *refusing* an answer to a query φ by weakening the correct answer to a *tautology* corresponds to returning $\varphi \cup (\mathcal{I} \setminus \varphi) = \mathcal{I}$, i.e., by saying that the actual instance might be any one, which a client is assumed to know anyhow. Furthermore *lying* on φ by *negation* corresponds to returning the complement $\mathcal{I} \setminus \varphi$ of the correct answer. Hence, as by the model theory of a logic, the intuitive meanings of phrases like "conjunction", "disjunction", "negation" or "tautology" are reflected by set-theoretic operations on sets of instances, i.e., by some algebra over the powerset of $\mathcal{I}$ (or a suitable subset of that powerset). Finally, an *abstract potential secret* is just given by an abstract query.

Result 3 (Effectiveness of Basic Censors in the Abstract Framework). *For the abstract framework used for controlling* sequences of queries *or controlled* view publishing*, respectively, the application of each of the basic censors for* refusal*,* lying*, or the* combination *of refusal and lying, respectively,* preserves confidentiality*.*

Result 4 (Refusal as Normal Form). *For the abstract framework used for controlled* view publishing*, the achievements of* any *effective censor can equivalently be described in terms of the basic* refusal *approach.*

Result 5 (Limits of Refusals are not Refinable). *For the abstract framework used for controlled* view publishing*, the limit (under intersection) of the*

controlled answers under the basic refusal *approach of any exhaustive query sequence cannot be refined, and in this sense it is* optimal.

Notably, for the basic *lying* approach a corresponding result does *not* hold, essentially due to the need of protecting disjunctions of potential secrets.

Result 6 (Inherent Computational Complexity of Optimal Censoring). *For the abstract framework used for controlled* view publishing *(and thus, in a sense, for any sufficiently expressive framework), under suitable assumptions about the finiteness of the situation and the encoding of censors and their inversions, the following problem is* coNP-complete*: given a confidentiality policy and a censor, decide whether for each instance of the information system the censor generates a published view that is both confidentiality preserving and optimal w.r.t. the policy.*

Issue 3 (Notions of Optimality and Related Approximations). *Define and investigate meaningful notions of* optimality*, capturing suitable intuitions of "best availability", together with convincing notions of* approximation *to overcome the inherently high computational complexity.*

So far, in the simple propositional framework as well as in its extensions and in the abstract framework, the works on CIE have considered a *possibilistic* notion of confidentiality, which only requires the *existence* of at least one witness of the required property regarding an alternative instance that is both indistinguishable and "harmless". However, one might be interested in a more refined notion which treats *degrees of confidentiality* based on an *evaluation* of all ìndistinguishable instances regarding being either "harmful" or "harmless" [60,61,69]. For example, such an evaluation might count the *cardinalities* of the two classes and then relate the cardinalities according to a declared threshold or, if an a priori probability distribution over the set of all instances is known, determine and relate the respective *probabilities*.

Issue 4 (Generalized Abstract Framework). *Generalize the abstract framework so far dealing with possibilistic confidentiality towards kinds of* evaluated confidentiality*, in particular probabilistic confidentiality.*

4 A Relational Framework

On the one hand, classical propositional logic over a finite set of propositional atoms, as considered in Sect. 2, enjoys many nice computational properties, including computationally solvable decision problems with theorem provers like SAT(atisfiability)-solvers and C(onstraint)S(atisfaction)P(roblem)-solvers which are usually highly efficient [42,62,67,83] (despite the intractable worst-case complexity). But on the other hand, that logic lacks expressiveness to capture many features needed for more advanced applications. In contrast, classical *first-order logic* is often expressive enough for such needs but suffers from essential restrictions regarding general decidability and from potentially unaffordable computational efficiency of decidable fragments [43] or practical theorem proving [71,79,80].

Serving as a foundation of *relational databases* [1], first-order logic provides formal means to interpret a stored relational instance as a (semantic) *model* satisfying the *integrity constraints* declared in the schema and to deal with *open queries*, intuitively of the kind "give me all $x, y, \ldots$ such that the property ... holds". In this context, an open query is expressed by a formula containing free occurrences of one or more variables and expected to return those sentences that result from substituting the free occurrences with constants and then, as a (closed) sentence, are evaluated to *true* regarding the stored relational instance.

However, a closer look reveals that we have to take care about several subtle details. Classical model theory for first-order logic deals with universes (sets over which interpretations are formed and variables are ranging) and with interpretations of relation symbols of *any cardinality* [68,75]. Accordingly, a formula with free variables might return infinitely many "true" sentences by substitution, even worse, without any further specification by typing, from any universe. But a database relation is a stored *finite* object, and an open query should always return a *finite* object, too. The latter property is guaranteed if the query formula is *domain-independent* and in particular *safe* (see, e.g., [1]), i.e., intuitively, whenever a *negation* occurs in the query formula – in principle evaluated by taking a set-theoretic complement w.r.t. to some previously determined and possibly infinite set – or a variable occurs – in principle ranging over a possibly infinite set – then the possibility of dealing with an infinite set does not actually occur, since the pertinent sets can be bounded to a finite subset.

Tentatively, all these problems could be avoided by employing only models with a *finite universe* and thus finite interpretations [52,64]. But then at least the following problems occur: applications often suggest not to define a cardinality bound on the type of an attribute in a relation scheme, and inference control often wants to avoid *combinatorial inferences* based on a fixed and known finite cardinality of some set, like applying the pigeon hole principle, in particular when the application does not justify such a bound.

Seeing neither the classical model-theoretic semantics nor the finite-model semantics as appropriate for general inference control of advanced applications, all works of CIE dealing with relational databases [16,18–20,22,23,25–27,35,38] are based on so-called *DB-semantics*: interpretations are restricted to Herbrand-like ones over a fixed *infinite* universe of *constant symbols*, which are constraint by *unique names* axioms, with only *finitely many positively evaluated ground* facts. This feature has some unusual consequences, e.g., for each safe open query formula $\varphi(x)$, the sentence $(\exists x)[\neg\varphi(x)]$ is a tautology. But a comprehensive exploration of the exact relationship between classical semantics, finite-model semantics and DB-semantics appears to be not available, but see, e.g., [1,3,16, 63,81].

Issue 5 (Logical Foundation of the Relational Model). *Reconsider the* theory of relational databases *in terms of first-order logic with* DB-semantics*, postulating* infinite domains *of constants used as* unique names *but considering only* finite relational instances.

Employing DB-semantics and restricting a priori knowledge including integrity constraints, confidentiality policies, (closed) query sentences and (open) query formulas such that all DB-entailment problems to be considered by a censor will be in a suitable decidable fragment of first-order logic, the basic approaches to construct a censor for sequences of queries, originally designed for closed queries only, can be extended to include also open queries [16,18]. More specifically, the extension is based on the decidability of the universal validity problem of the *Bernays-Schoenfinkel class* of sentences in prenex normal form having an $\forall^*\exists^*$ prefix, which not only holds for classical semantics and finite-model semantics, but also for DB-semantics.

The extensions of the basic approaches are then based on the following features, the first and the second of which are supported by DB-semantics:

- An open query can be evaluated by systematically *enumerating all substitutions* of the free occurrences of variables in the query formula by constants taking from the fixed universe, and handling the resulting sentences as *closed queries* to be controlled.
- Such an in principle infinite enumeration can be *terminated* after a finite number of rounds by suitably inspecting pertinent *completeness sentences* that basically state that in all further rounds the considered closed queries will be answered negatively, basically capturing a *closed world assumption* for the answers generated before. As far as needed, and at least after termination, the *controlled* truth evaluation of such a completeness sentence is explicitly added to the current view, and thus any implicit knowledge provided by the closed-world assumption is under effective inference control.
- The pertinent completeness sentences are expressible in first-order logic such that their usage in the entailment problems inspected by the censor remains in the decidable fragment.
- Besides others, statically *fixing* the enumeration sequence in advance ensures the kind of indistinguishability required by the formal notion of preservation of confidentiality, even if that enumeration is known to the client.

Result 7 (Effectiveness of Basic Approaches for Query Sequences). *For the relational framework under* DB-semantics *of first-order logic used for controlling* sequences of queries *including open ones, each of the basic approaches of* refusal, lying, *or the* combination *of refusal and lying, respectively, can be extended to* open queries. *In each case, the extended censor controls sufficiently many closed sentences obtained by a substitution in a fixed sequence and inspects suitably formed completeness sentences in a controlled way, such that each controlled answer processing* terminates *and* preserves confidentiality.

Issue 6 (Entailment Problems with Completeness Sentences). *Explore efficient computational approaches to decide* entailment problems *of first-order logic under DB-semantics when relational* completeness sentences *are involved.*

5 Static View Publishing

Research on confidentiality-preserving *view publishing* [57,58] spans a broad range of frameworks, including pioneering work on distortions of *statistical databases* [50,82], *value generalization* and *row-suppressing* for achieving *k-anonymity* and *l-diversity* of tables [46,66], and *database fragmentation and encryption* for cloud computing [2,45,48,59]. View publishing has also been studied for CIE for several frameworks and approaches to censor construction, guided by three different strategies as discussed below:

1. for the abstract framework using any of the basic approaches, by taking the limit of controlled answers to an exhaustive sequence of all queries [17];
2. somehow implicitly, for the relational framework with DB-semantics following any of the basic approaches, by controlling those open queries that would return a whole relation, based on a fixed exhaustive sequence of all closed and elementary queries each of which is about just one tuple [16];
3. for a specific description logics framework [4] using a variant of the basic approaches, by iteratively enumerating all possible atoms of the logic [40,41];
4. for both the propositional and the relational framework with DB-semantics following the lying approach, by iteratively modifying a given instance while also aiming at a minimum number of distortions [19,37,38];
5. for an XML-approach following a weakening approach by iteratively suppressing harmful parts [24];
6. for the relational framework with DB-semantics following a weakening approach that refines the refusal approach by globally determined value generalization [26]; and
7. for the relational framework with DB-semantics following a weakening approach by globally determined fragmentation and encryption [25,27].

The first kind of a strategy [16,17,40,41], items 1–3 above, treats a view in an *intensional* way, seeing a view as being fully characterized by its relevant properties. In this context, the relevant properties are the *controlled answers* to an *exhaustive sequence of queries* evaluated regarding the actual instance. In the abstract framework [17], see Sect. 3, due to the lack of any internal structure of instances, exhaustiveness requires to include *all* queries. In the relational framework [16], see Sect. 4, where an instance is built from tuples, exhaustiveness can be accomplished by including all *elementary* queries about just one tuple. Similarly, in the description logics framework [40,41] all atoms are employed. In more procedural terms, the view to be published is iteratively approximated "from above", starting with full ignorance (or with the assumed a priori knowledge) and then stepwise adding information to narrow it down towards the final limit. And in computational terms, the iteration should terminate in finite time to come up with a final view.

The second kind of a strategy [19,24,37,38], items 4–5 above, works in an *extensional* way, starting with the extension of the actual instance and treating both the elements of the a priori knowledge and the potential secrets in the confidentiality policy as *constraints*, employed for iteratively *modifying* the original

instance: as long as any of the given or dynamically derived constraints is still violated, a violating constraint is selected and the currently considered instance is minimally modified to comply with the selected constraint. So, the view to be published is approximated "from below", starting with the actual instance and then stepwise distorting it. Again, the crucial point is termination: a modification to satisfy one constraint might cause to newly violate another one. Clearly, if the framework is expressive enough, the *constraint satisfaction problem* becomes undecidable, and thus we have to suitably restrict the expressiveness.

A third kind of strategy [25–27], items 6–7 above, also works in an *extensional* way, but in a sense more globally than iteratively. Regarding [26], without giving details here, in a first instance-independent step only considering the potential secrets, some kind of constraints on "admissible" weakenings are generated, which then, at least conceptually, are "globally solved" in a minimal way (where the actually used solver might work sequentially). Only in a second step, the actual relational instance is weakened by converting each harmful tuple (in logic terms, each ground fact) into an admissible disjunction. This two step procedure ensures that undistorted parts of the view remain isolated from weakened parts, and thus any harmful inferences are blocked. A related guarantee by isolation is employed in [25,27]. Again, computability and efficiency is a problem, demanding for suitable restrictions.

In all strategies, while giving precedence to preserve confidentiality, availability is considered as an important secondary goal. Accordingly, the "difference" between the actual instance and the view to be published should be at least "minimal" in the sense that discarding any single distortion would lead to a violation of confidentiality. More ambitiously, however, we might even aim at finding a view that has a minimum number of distortions among the set of all confidentiality-preserving views.

So far, adding such an overall *numerical optimization problem* to the problem of preserving confidentiality has only been thoroughly treated for the relational framework following the lying approach with the extensionally working strategy [19,37,38]. Though this attempt has required to combine the *satisfaction problem* for sentences in an expressive fragment of first-order logic with a numerical *optimization problem*, it has been proved to be conceptually successful [38]. But this attempt appears to be not practically feasible in general, and thus often requires to relax the optimization requirement by allowing an approximation or to suitably restrict the constraints [19].

Result 8 (Intensional Iterative View Generation by Exhaustive Querying). *Subject to appropriate operations of information manipulation and to termination, an* intensionally *working and* iteratively *proceeding generation strategy returns a* view *that* preserves confidentiality.

Result 9 (Extensional Iterative View Generation by Eliminating Violations). *Subject to appropriate restrictions on expressiveness and to termination, an* extensionally *working and* iteratively *proceeding generation strategy returns a* view *that* preserves confidentiality.

Result 10 (Extensional View Generation by Global Distortions). *Subject to appropriate restrictions on expressiveness, in dedicated cases an* extensionally *working generation strategy that globally determines distortions returns a* view *that* preserves confidentiality.

Issue 7 (Comparison of Generalized View Generation Strategies). *Generalize and elaborate both the extensionally working and the intensionally working view generation strategy, respectively, and systematically compare their achievements, in particular regarding the availability of information provided by the confidentiality-preserving views.*

6 Advanced Reasoning

In both the propositional framework of Sect. 2 and the relational framework of Sect. 4, the underlying logic-oriented information system is supposed to *completely* describe some outside "real world" by storing a representation of a (semantic) model which assigns a truth value to *all* atomic sentences and thus, by induction, to *all* sentences. In many applications, however, the (owner of the) information system might have only *incomplete knowledge* about the outside world or even only some *fragmentary internal belief*. In the rich literature about knowledge and belief engineering, many approaches to deal with such situations have been proposed and studied in detail, see, e.g., [4,6,44,53,56,63].

For inference control by means of CIE, *incompleteness* has first been treated for an extended propositional framework [36]: now, an *instance db* of the information system is a consistent set of propositional sentences of the language $\mathcal{L}_{pl}$ of classical propositional logic over a finite set of propositional atoms. While, syntactically, a *query* φ is still a sentence of $\mathcal{L}_{pl}$, semantically its evaluation is now based on the notion of entailment, also denoted by $\models$, rather than directly on truth evaluation with respect to a (semantic) model, tentatively given by

$$eval(db, \varphi) := \begin{cases} true & \text{if } db \models \varphi, \\ false & \text{if } db \models \neg\varphi, \\ undefined & \text{otherwise.} \end{cases} \tag{5}$$

whereas as before the definite results of the first two cases are directly expressible in $\mathcal{L}_{pl}$, the result of the third case is not. So, extending propositional logic, a *knowledge operator* K for a *modal logic* [53] is introduced to enable us to speak about "the information system knows that ..." and, correspondingly, "the information system does not know that ...":

$$eval(db, \varphi) := \begin{cases} K\varphi & \text{if } db \models \varphi, \\ K\neg\varphi & \text{if } db \models \neg\varphi, \\ \neg K\varphi \wedge \neg K\neg\varphi & \text{otherwise.} \end{cases} \tag{6}$$

By this approach, constructing a censor, we can now distinguish whether the information system itself does *not know* the answer to a query or whether the

censor merely demands to *refuse* an informative answer. More generally, we now have four possible controlled answers, which provides additional flexibility for distorting answers. This flexibility is exploited by defining so-called *distortion tables* which determine for each combination of a client state in need of a distortion and the correct answer a controlled (possibly distorted) "harmless" answer, based on a finite list of representations of the relevant client states.

Result 11 (Effectiveness of Adapted Basic Censors for Query Sequences to Incomplete Information Systems). *For the extended* propositional *framework with* incomplete *instances used for controlling* sequences of queries *based on modal logic and employing a* distortion table*, all adaptions of each of the basic censors for* refusal*,* lying*, or the* combination *of refusal and lying, respectively,* preserve confidentiality*.*

Whereas propositional modal logic evolves from classical propositional logic in a quite natural way, extending classical first-order logic by modalities requires highly sophisticated considerations [55]. So far our attempts to transform and extend the propositional case treated in [36] to the general first-order case, which among others have also been inspired by [63,70], have not been successful.

Issue 8 (First-Order Modal Logic for Censor Construction). *Elaborate the* modal logic approach *to construct censors for* incomplete instances *of an information system based on* first-order logic*.*

However, restricting the first-order case to a *finite situation*, we could successfully treat a comprehensive *propositionalization* [35].

Result 12 (Propositionalized First-Order Modal Logic for Censor Construction). *The* modal logic approach *to construct confidentiality-preserving censors for* incomplete instances *of an information system can be extended to a first-order logic framework that can be finitely propositionalized.*

An alternative way to deal with inference control for incomplete information systems [40,41] has been based on description logics, which provides efficiently tractable fragments of first-order logic.

Result 13 (Censor Construction for Incomplete Information Systems Based on Description Logics). *For a* description logics *framework of an* incomplete *information system used for controlled* view publishing *following a variant of the basic approaches, the limit of the controlled answers of any exhaustive sequence of atoms* preserves confidentiality*.*

Though not elaborated in the context of CIE, a further interesting and very flexible approach to censor constructions for sequences of queries evaluated w.r.t. an incomplete information system has been proposed for a Boolean description logics framework [78].

Incompleteness of an instance complicates query answering and view publishing by the information system, and thus also increases the client's challenge

to infer confidential information from visible reaction data. Though the client does not know the incomplete instance stored by the information system, the client is still fully aware about the system's reasoning procedure to generate an answer or a view, respectively.

However, the situation is changing, and becoming even more challenging, if the information system represents an internal *belief*, which is not only based on (classical) sentences but also on *conditionals* (also known as default rules or probabilistic rules). In order to form a consistent belief, such a system employs *non-monotonic reasoning* parameterized with an *instantiation* of some *plausibility structure* such as preference orderings, ordinal conditional functions, possibility or plausibility spaces [56]. The client then faces the additional problem of being *uncertain* about the concrete instantiation actually used by the system, and thus also the censor has to appropriately deal with that uncertainty.

Exemplarily for potentially many similar situations, CIE has been conceptually extended for a propositional information system that is based on *ordinal conditional functions* [6,77] – or, more generally, an abstract class of suitable consequence relations – and handles query requests regarding its current belief as well as revision requests [30,33]. In this work, such an abstract class is shown to be obtainable by an "allowed" axiomatization, and the censor construction is directed to preserve confidentiality regarding a client that knows the pertinent class and masters its uncertainty about which instantiation is taken by "accepting" a sentence if and only if the sentence is plausible under *all* instantiations. But other kinds of treating that kind of uncertainty could also be meaningful, for instance credulous reasoning.

Result 14 (Effectiveness of a Refusal Censor for Sequences of Belief Queries and Belief Revisions). *For a non-monotonic propositional framework for* belief *based on a* class of consequence relations *having an "allowed" axiomatization used for controlling* mixed sequences of queries and revisions*, a computational adaption of the basic censor for* refusal preserves confidentiality *assuming a* skeptically *reasoning client.*

Issue 9 (General Censor Constructions for Non-monotonic Frameworks). *For further examples of a non-monotonic framework, explore the options to construct a confidentiality-preserving censor, and concisely generalize such constructions.*

A crucially important aspect of any censor construction for an information system based on advanced reasoning is an (at least) two-step reflection of the system's reasoning under mutual uncertainty. As further discussed in [34], such a reflection is needed for the client simulator in the rough architecture of CIE shown in Fig. 1.

Issue 10 (A Censor's Simulation of the Client's Inference of the System's Parameterized Belief). *Given an information system based on advanced* parameterized belief reasoning*, identify the following: (i) what a client*

can infer about the system's belief from the reactions and (ii) what a censor can computationally determine about what the client can infer, for both cases of whether the reactions are controlled or not, respectively.

7 Advanced Interactions

Early work about CIE has focused on inference control of query answering and view publishing regarding a *fixed* instance stored by the underlying information system. In general, however, an instance will be *modified* over the time. Then answers to queries become time-dependent, and a simple syntactic view obtained by the client by directly logging the data received might become inconsistent. Moreover, not only the information system can autonomously modify the instance, but the client itself might request a modification. For example, in a multiagent system, after having observed that the outside "real world" has changed, a client agent might inform the information system agent about the observation and request a corresponding *update* of the system's belief. Or a client agent has learnt further aspects about the unchanged "real world" and then suggests a corresponding *revision* of the system's belief.

In general, processing an update or revision request follows a sometimes quite involved protocol, in particular in order to maintain *invariants* declared in the schema of the information system in the form of sentences expressing *integrity constraints*, which are seen as being "unmodifiable" or, in other terms, "unquestionable". If a requested modification would violate a constraint, the request is either totally rejected or at least somehow "corrected". In any case, the information system would externally react by sending a corresponding notification to the client. Moreover, some complicated updates or revisions can only be handled as *transactions* such that only the finally resulting instance is guaranteed to satisfy the constraints, but the auxiliary versions generated during processing are not.

Now, receiving a *notification* about success, correction or failure of a modification request implies getting answers to *implicit queries* regarding the constraints. Thus inference control of interactions that modify the instance has to suitably distort such notifications in order to enforce the required confidentiality. Unfortunately, early research on multilevel databases with mandatory access control has already shown that maintaining integrity on the one hand and enforcing confidentiality on the other hand might be conflicting goals [47,49,65,73]. A proposed resolution has been the concept of *polyinstantiation*, i.e., introducing some kind of cover stories or lies for specific clients.

If inference control considers that a client infers knowledge in a *history-aware* way, as CIE is doing, a further difficulty arises. Observing time-dependent data about different versions of the stored instance, the client might get new options of inferences by reasoning about the causes that led to semantically different reactions on syntactically the same or related requests. Moreover, later reactions might reveal that confidential information has been valid earlier. Thus, if wanted according to the application, *continuous confidentiality preservation* might be required, i.e., to not only confine knowledge about the current instance but also about previous ones.

The problems sketched above have first been studied for the propositional framework presented in Sect. 2, suitably extended to include single updates as well as transactional ones under some simplifying restrictions [21,29].

Result 15 (Effectiveness of Adapted Basic Censors for Sequences of Queries and Updates). *For the* propositional *framework with* complete *instances used for controlling mixed sequences of* queries *and suitably restricted single or transactional* updates*, adaptions of the basic censors for* refusal *or* lying*, respectively,* preserve confidentiality*.*

As mentioned before in Result 14, a similar achievement has been obtained for a propositional framework with belief revision [32].

Issue 11 (General Censor Constructions for Classical, Incomplete and Non-monotonic Frameworks with Modification of Instances). *For further examples of a classical, incomplete or non-monotonic framework with updates and, as far as applicable, revisions, explore the options to construct a confidentiality-preserving censor, and concisely generalize these constructions.*

A protocol for processing a modification can be seen as a *procedural application program* or a *stored procedure* that, depending on the client's request,

- generates and submits queries regarding the current instance,
- potentially level-wise branches according to the corresponding answers used as conditions in guarded commands, and
- in each branch
  - actually modifies the instance in a possibly "corrected" way,
  - prepares a corresponding notification and
  - finally sends it to the client.

Clearly, such procedural programs are of interest not only for specific processing of modifications but for reacting on any kind of messages received from a client. So, we would like to elaborate a generic approach to apply inference control for the execution of any such procedural program, in particular for preparing controlled notifications. Accordingly, still under some restrictions, in recent and ongoing work [31,32] we have designed and verified a combination

- of CIE-like inference control by means of *abstract representations* of the information content of program variables keeping answers to queries regarding the stored instance and of suitably generated *distortion tables*
- with security techniques for language-based *information flow control*, in particular capturing *implicit flows* by guarded commands, by means of *security typing* and of *declassification* [5,72].

Result 16 (Controlled Mediation of Client Requests Processed by Procedural Programs). *Assuming an integrated* fixed *belief instance obtained from one or more underlying information systems (and thus so far* not *allowing modifications of that belief and, suitably propagated, of the underlying instances),*

and restricting to guarded commands of the if-then-else *form (and thus so far* not *allowing arbitrary repetitions) and to sensitive program variables with manageably* small domain *extensions, the designed combination of CIE-like inference control following a* weakening *approach with language-based information flow control* preserves confidentiality*.*

Issue 12 (Generalized Controlled Mediation of Client Requests Processed by Procedural Programs). *Extend and generalize the designed combination of* CIE-like inference control *with* language-based information flow control *for* procedural programs *as expressive as possible.*

Acknowledgements. I would like to sincerely thank all colleagues who have worked together with me on Controlled Interaction Execution, in particular the co-authors of joint publications. Moreover, I am specially indebted to Marcel Preuß and Cornelia Tadros for many helpful comments on an earlier draft. Finally, I gratefully acknowledge the longtime support of the German Research Council, DFG, under grants Bi 311/12 and SFB 876/A5.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: a distributed architecture for secure database services. In: 2nd Biennial Conference on Innovative Data Systems Research, CIDR 2005, pp. 186–199. Online Proceedings (2005)
3. Ailamazyan, A.K., Gilula, M.M., Stolbushkin, A.P., Shvarts, G.F.: Reduction of a relational model with infinite domains to the finite-domain case. Russian version: Dokl. Akad. Nauk SSSR **286**, 308–311; English translation: Sov. Phys. Dokl. **31**(1), 11–13 (1968)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
5. Balliu, M., Dam, M., Guernic, G.L.: Encover: symbolic exploration for information flow security. In: Chong, S. (ed.) IEEE Computer Security Foundations Symposium, CSF 2012, pp. 30–44. IEEE Computer Society, Los Alamitos (2012)
6. Beierle, C., Kern-Isberner, G.: A conceptual agent model based on a uniform approach to various belief operations. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009. LNCS, vol. 5803, pp. 273–280. Springer, Heidelberg (2009)
7. Bell, D.E., LaPadula, L.J.: Secure computer systems: a mathematical model, volume II. J. Comput. Sec. **4**(2/3), 229–263 (1996). Reprint of MITRE Corporation (1974)
8. Biskup, J.: For unknown secrecies refusal is better than lying. Data Knowl. Eng. **33**(1), 1–23 (2000)
9. Biskup, J.: Security in Computing Systems - Challenges. Approaches and Solutions. Springer, Heidelberg (2009)

10. Biskup, J.: Dynamic policy adaption for inference control of queries to a propositional information system. J. Comput. Secur. **20**, 509–546 (2012)
11. Biskup, J.: Inference-usability confinement by maintaining inference-proof views of an information system. Int. J. Comput. Sci. Eng. **7**(1), 17–37 (2012)
12. Biskup, J.: Logic-oriented confidentiality policies for controlled interaction execution. In: Madaan, A., Kikuchi, S., Bhalla, S. (eds.) DNIS 2013. LNCS, vol. 7813, pp. 1–22. Springer, Heidelberg (2013)
13. Biskup, J., Bonatti, P.A.: Lying versus refusal for known potential secrets. Data Knowl. Eng. **38**(2), 199–222 (2001)
14. Biskup, J., Bonatti, P.A.: Controlled query evaluation for enforcing confidentiality in complete information systems. Int. J. Inf. Secur. **3**(1), 14–27 (2004)
15. Biskup, J., Bonatti, P.A.: Controlled query evaluation for known policies by combining lying and refusal. Ann. Math. Artif. Intell. **40**(1–2), 37–62 (2004)
16. Biskup, J., Bonatti, P.A.: Controlled query evaluation with open queries for a decidable relational submodel. Ann. Math. Artif. Intell. **50**(1–2), 39–77 (2007)
17. Biskup, J., Bonatti, P.A., Galdi, C., Sauro, L.: Optimality and complexity of inference-proof data filtering and CQE. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014, Part II. LNCS, vol. 8713, pp. 165–181. Springer, Heidelberg (2014)
18. Biskup, J., Bring, M., Bulinski, M.: Confidentiality preserving evaluation of open relational queries. In: Morzy, T., Valduriez, P., Bellatreche, L. (eds.) ADBIS 2015. LNCS, vol. 9282, pp. 431–445. Springer, Heidelberg (2015)
19. Biskup, J., Dahn, C., Diekmann, K., Menzel, R., Schalge, D., Wiese, L.: Publishing inference-proof relational data: an implementation and experiments (2015) (submitted for publication)
20. Biskup, J., Embley, D.W., Lochner, J.H.: Reducing inference control to access control for normalized database schemas. Inf. Process. Lett. **106**(1), 8–12 (2008)
21. Biskup, J., Gogolin, C., Seiler, J., Weibert, T.: Inference-proof view update transactions with forwarded refreshments. J. Comput. Secur. **19**, 487–529 (2011)
22. Biskup, J., Hartmann, S., Link, S., Lochner, J.-H.: Efficient inference control for open relational queries. In: Foresti, S., Jajodia, S. (eds.) Data and Applications Security and Privacy XXIV. LNCS, vol. 6166, pp. 162–176. Springer, Heidelberg (2010)
23. Biskup, J., Hartmann, S., Link, S., Lochner, J.-H., Schlotmann, T.: Signature-based inference-usability confinement for relational databases under functional and join dependencies. In: Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J. (eds.) DBSec 2012. LNCS, vol. 7371, pp. 56–73. Springer, Heidelberg (2012)
24. Biskup, J., Li, L.: On inference-proof view processing of XML documents. IEEE Trans. Dependable Sec. Comput. **10**(2), 99–113 (2013)
25. Biskup, J., Preuß, M.: Database fragmentation with encryption: under which semantic constraints and a priori knowledge can two keep a secret? In: Wang, L., Shafiq, B. (eds.) DBSec 2013. LNCS, vol. 7964, pp. 17–32. Springer, Heidelberg (2013)
26. Biskup, J., Preuß, M.: Inference-proof data publishing by minimally weakening a database instance. In: Prakash, A., Shyamasundar, R. (eds.) ICISS 2014. LNCS, vol. 8880, pp. 30–49. Springer, Heidelberg (2014)
27. Biskup, J., Preuß, M., Wiese, L.: On the inference-proofness of database fragmentation satisfying confidentiality constraints. In: Lai, X., Zhou, J., Li, H. (eds.) ISC 2011. LNCS, vol. 7001, pp. 246–261. Springer, Heidelberg (2011)
28. Biskup, J., Tadros, C.: Policy-based secrecy in the Runs & Systems framework and controlled query evaluation. In: Echizen, I., Kunihiro, N., Sasaki, R. (eds.)

Advances in Information and Computer Security, IWSEC 2010, Short Papers, pp. 60–77. Information Processing Society of Japan (IPSJ) (2010)
29. Biskup, J., Tadros, C.: Inference-Proof View Update Transactions with Minimal Refusals. In: Garcia-Alfaro, J., Navarro-Arribas, G., Cuppens-Boulahia, N., de Capitani di Vimercati, S. (eds.) DPM 2011 and SETOP 2011. LNCS, vol. 7122, pp. 104–121. Springer, Heidelberg (2012)
30. Biskup, J., Tadros, C.: Revising belief without revealing secrets. In: Lukasiewicz, T., Sali, A. (eds.) FoIKS 2012. LNCS, vol. 7153, pp. 51–70. Springer, Heidelberg (2012)
31. Biskup, J., Tadros, C.: Confidentiality enforcement by hybrid control of flows from abstract information states through program execution via declassification (2015) (submitted for publication)
32. Biskup, J., Tadros, C.: Constructing inference-proof belief mediators. In: Samarati, P. (ed.) DBSec 2015. LNCS, vol. 9149, pp. 188–203. Springer, Heidelberg (2015)
33. Biskup, J., Tadros, C.: Preserving confidentiality while reacting on iterated queries and belief revisions. Ann. Math. Artif. Intell. **73**(1–2), 75–123 (2015)
34. Biskup, J., Tadros, C.: On the simulation assumption for controlled interaction processing (to appear, 2016)
35. Biskup, J., Tadros, C., Wiese, L.: Towards controlled query evaluation for incomplete first-order databases. In: Link, S., Prade, H. (eds.) FoIKS 2010. LNCS, vol. 5956, pp. 230–247. Springer, Heidelberg (2010)
36. Biskup, J., Weibert, T.: Keeping secrets in incomplete databases. Int. J. Inf. Secur. **7**(3), 199–217 (2008)
37. Biskup, J., Wiese, L.: Preprocessing for controlled query evaluation with availability policy. J. Comput. Secur. **16**(4), 477–494 (2008)
38. Biskup, J., Wiese, L.: A sound and complete model-generation procedure for consistent and confidentiality-preserving databases. Theoret. Comput. Sci. **412**, 4044–4072 (2011)
39. Bonatti, P.A., Kraus, S., Subrahmanian, V.S.: Foundations of secure deductive databases. IEEE Trans. Knowl. Data Eng. **7**(3), 406–422 (1995)
40. Bonatti, P.A., Petrova, I.M., Sauro, L.: Optimized construction of secure knowledge-base views. In: Calvanese, D., Konev, B. (eds.) International Workshop on Description Logics 2015. CEUR Workshop Proceedings, vol. 1350. CEUR-WS.org (2015)
41. Bonatti, P.A., Sauro, L.: A confidentiality model for ontologies. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 17–32. Springer, Heidelberg (2013)
42. Bordeaux, L., Hamadi, Y., Zhang, L.: Propositional satisfiability and constraint programming: a comparative survey. ACM Comput. Surv. **38**(4), 12.1–12.54 (2006)
43. Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. Perspectives in Mathematical Logic. Springer, Heidelberg (1997)
44. Brachman, R.J., Levesque, H.J.: Knowledge Representation and Reasoning. Elsevier, Amsterdam (2004)
45. Ciriani, V., De Capitani di Vermercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. ACM Trans. Inf. Syst. Secur. **13**(3), 1–33 (2010)
46. Ciriani, V., De Capitani di Vermercati, S., Foresti, S., Samarati, P.: K-anonymity. In: Yu, T., Jajodia, S. (eds.) Secure Data Management in Decentralized Systems. Advances in Information Security, vol. 33, pp. 323–353. Springer, New York (2007)

47. Cuppens, F., Gabillon, A.: Cover story management. Data Knowl. Eng. **37**(2), 177–201 (2001)
48. De Capitani di Vermercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Fragmentation in presence of data dependencies. IEEE Trans. Dependable Sec. Comput. **11**(6), 510–523 (2014)
49. Denning, D.E., Akl, S.G., Heckman, M., Lunt, T.F., Morgenstern, M., Neumann, P.G., Schell, R.R.: Views for multilevel database security. IEEE Trans. Software Eng. **13**(2), 129–140 (1987)
50. Denning, D.E., Schlörer, J.: Inference controls for statistical databases. IEEE Comput. **16**(7), 69–82 (1983)
51. Dolzhenko, E., Ligatti, J., Reddy, S.: Modeling runtime enforcement with mandatory results automata. Int. J. Inf. Secur. **14**(1), 47–60 (2015)
52. Ebbinghaus, H.D., Flum, J.: Finite Model Theory. Springer, Heidelberg (1995)
53. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
54. Farkas, C., Jajodia, S.: The inference problem: a survey. SIGKDD Explor. **4**(2), 6–11 (2002)
55. Fitting, M., Mendelsohn, R.L.: First-Order Modal Logic, Synthese Library, vol. 277. Kluwer Academic Publishers, Dordrecht (1998)
56. Friedman, N., Halpern, J.Y.: Plausibility measures and default reasoning. J. ACM **48**(4), 648–685 (2001)
57. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: a survey of recent developments. ACM Comput. Surv. **42**(4), 1–53 (2010)
58. Fung, B.C.M., Wang, K., Fu, A.W.C., Yu, P.S.: Introduction to Privacy-Preserving Data Publishing - Concepts and Techniques. Chapman & Hall/CRC, Boca Raton (2010)
59. Ganapathy, V., Thomas, D., Feder, T., Garcia-Molina, H., Motwani, R.: Distributing data for secure database services. Trans. Data Priv. **5**(1), 253–272 (2012)
60. Gray III, J.W.: Toward a mathematical foundation for information flow security. In: IEEE Symposium on Security and Privacy, pp. 21–35 (1991)
61. Halpern, J.Y., O'Neill, K.R.: Secrecy in multiagent systems. ACM Trans. Inf. Syst. Secur. **12**(1), 1–47 (2008)
62. Katebi, H., Sakallah, K.A., Marques-Silva, J.P.: Empirical study of the anatomy of modern sat solvers. In: Sakallah, K.A., Simon, L. (eds.) SAT 2011. LNCS, vol. 6695, pp. 343–356. Springer, Heidelberg (2011)
63. Levesque, H.J., Lakemeyer, G.: The Logic of Knowledge Bases. MIT Press, Cambridge (2000)
64. Libkin, L.: Elements of Finite Model Theory. Springer, Heidelberg (2004)
65. Lunt, T.F., Denning, D.E., Schell, R.R., Heckman, M., Shockley, W.R.: The SeaView security model. IEEE Trans. Software Eng. **16**(6), 593–607 (1990)
66. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: *L*-diversity: privacy beyond k-anonymity. TKDD **1**(1), 3 (2007)
67. Malik, S., Zhang, L.: Boolean satisfiability from theoretical hardness to practical success. Commun. ACM **52**(8), 76–82 (2009)
68. Nerode, A., Shore, R.: Logic for Applications, 2nd edn. Springer, New York (1997)
69. Ray, D., Ligatti, J.: A theory of gray security policies. In: Pernul, G., Ryan, P.Y.A., Weippl, E.R. (eds.) ESORICS 2015. LNCS, vol. 9327, pp. 481–499. Springer, Heidelberg (2015)
70. Reiter, R.: What should a database know? Logic Program. **14**, 127–153 (1992)
71. Robinson, J.A., Voronkov, A. (eds.): Handbook of Automated Reasoning (in 2 volumes). Elsevier, MIT Press, Amsterdam, Cambridge (2001)

72. Sabelfeld, A., Sands, D.: Dimensions and principles of declassification. In: IEEE Computer Security Foundations Workshop, CSFW 2005, pp. 255–269. IEEE Computer Society (2005)
73. Sandhu, R.S., Jajodia, S.: Polyinstantation for cover stories. In: Deswarte, Y., Quisquater, J.-J., Eizenberg, G. (eds.) ESORICS 1992. LNCS, pp. 307–328. Springer, Heidelberg (1992)
74. Schneider, F.B.: Enforceable security policies. ACM Trans. Inf. Syst. Secur. **3**(1), 30–50 (2000)
75. Shoenfield, J.R.: Mathematical Logic. Addison-Wesley, Reading (1967)
76. Sicherman, G.L., de Jonge, W., van de Riet, R.P.: Answering queries without revealing secrets. ACM Trans. Database Syst. **8**(1), 41–59 (1983)
77. Spohn, W.: Ordinal conditional functions: A dynamic theory of epistemic states. In: Skyrms, B., Harper, W.L. (eds.) Irvine Conference on Probability and Causation. Causation in Decision, Belief Change, and Statistics, vol. II, pp. 105–134. Kluwer, Dordrecht (1988)
78. Studer, T., Werner, J.: Censors for boolean description logic. Trans. Data Priv. **7**(3), 223–252 (2014)
79. Sutcliff, G., Suttner, C.: The TPTP problem library for automated theorem proving. Technical report (2015). http://www.tptp.org
80. Sutcliffe, G.: The TPTP problem library and associated infrastructure: The FOF and CNF parts, v3.5.0. J. Autom. Reason. **43**(4), 337–362 (2009)
81. Thalheim, B.: Entity-Relationship Modeling - Foundations of Database Technology. Springer, Heidelberg (2000)
82. Traub, J.F., Yemini, Y., Wozniakowski, H.: The statistical security of a statistical database. ACM Trans. Database Syst. **9**(4), 672–679 (1984)
83. Weissenbacher, G., Malik, S.: Boolean satisfiability solvers: techniques and extensions. In: Nipkow, T., Grumberg, O., Hauptmann, B. (eds.) Software Safety and Security - Tools for Analysis and Verification, pp. 205–253. IOS Press (2012)