Logic-Oriented Confidentiality Policies for Controlled Interaction Execution*

Joachim Biskup

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

Abstract. Controlled Interaction Execution is a specific concept of inference control for logic-oriented information systems. Interactions include query answering, update processing and data publishing, and operational control is based on declarative policies. Complementing a previous survey on this concept, in this work we treat various forms of confidentiality policies regarding their syntax, semantics, algorithms and pragmatics.In each case, we consider an information provider's interest in confidentiality as an exception from his general willingness and agreement to share information with a client.

Keywords: availability, confidentiality policy, continuous confidentiality, controlled interaction execution, data publishing, disjunctive policy element, epistemic potential secret, indistinguishability, inference control, inference-proof view, inference-usability confinement, information sharing, policy adaptation, possibilistic secrecy, potential secret, query answering, secrecy, temporary confidentiality, update processing.

1 Introduction

Inference-usability confinement is a specific kind of inference control [3, 32, 34], solely performed by a data provider without intervening at the site of a client. The confinement aims at customizing potentially sensitive data to be returned to an intelligent client such that the manipulated items (1) are still informative for the recipient but (2) do not enable him to gain more information than intended. The concept of Controlled Interaction Execution (CIE) [1, 4, 6–9, 11, 12, 16, 23, 26–28, 30, 36, 39, 44] explores a variety of algorithmic means to implement inference-usability confinement for logic-oriented information systems under query answering, update processing and data publishing.

Within this concept, the intended usage of data by a client is described by two complementary items: on the one hand, a presupposed general permission to share information according to some application-dependent agreement; on the other hand, specific exceptions referring to information to be kept confidential

* This work has been supported by the Deutsche Forschungsgemeinschaft (German Research Council) under grant SFB 876/A5 within the framework of the Collaborative Research Center "Providing Information by Resource-Constrained Data Analysis".

A. Madaan, S. Kikuchi, and S. Bhalla (Eds.): DNIS 2013, LNCS 7813, pp. 1–22, 2013.

according to individual or legal requirements. These exceptions are formally captured by a *confidentiality policy*:

- *Syntactically*, the declaration of a *policy instance* just specifies a set of sensitive pieces of information to be protected against some client who is suspected to be "too curious".
- *Semantically*, however, such a declaration requires a sophisticated invariant to be ensured over the time, for any possible sequence of interactions: as a data provider, the underlying information system should employ a control mechanism to confine the information content of all data returned to a client in such a way that the client will never be able to acquire the sensitive information – neither directly, nor by inferences based on a priori knowledge, data returned to him, and awareness of the control mechanism. Such an invariant may have different formalizations for different situations and, accordingly, in each case appropriate *policy semantics* have to be selected.
- *Algorithmically*, a policy has to be stepwise enforced by some appropriate *censor* as the main component of the control mechanism: at each individual point in time and for each single interaction request, that censor has to inspect the specific request and the current situation for compliance with the overall goal of the policy and to generate a suitable reaction.
- *Pragmatically*, a security officer should follow tool-supported business rules to determine a policy instance together with the policy semantics and an appropriate censor for any specific application.

Complementing a previous survey on CIE [2, 5], see also Figure 1, in this report we discuss the various forms of confidentiality policies regarding their syntax, semantics, algorithms and pragmatics. In particular, we list and summarize fundamental insight gained so far and point to challenging problems. Figure 2 illustrates the main features by an Entity-Relationship model. As a disclaimer, our contribution concentrates on the fundamental concepts and their formalizations rather than on empirical studies.

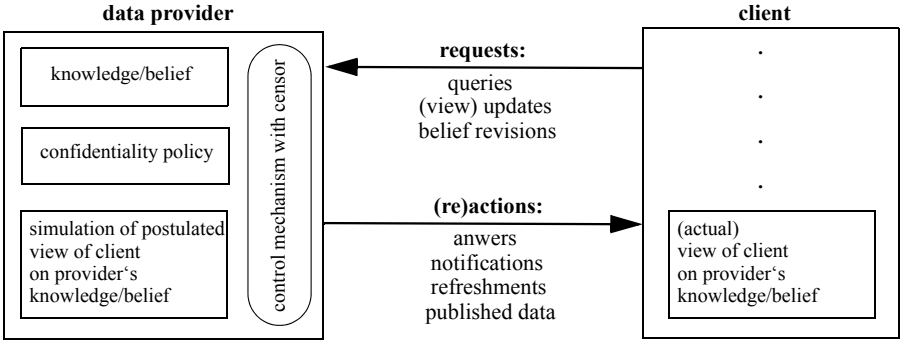

Fig. 1. General framework of CIE

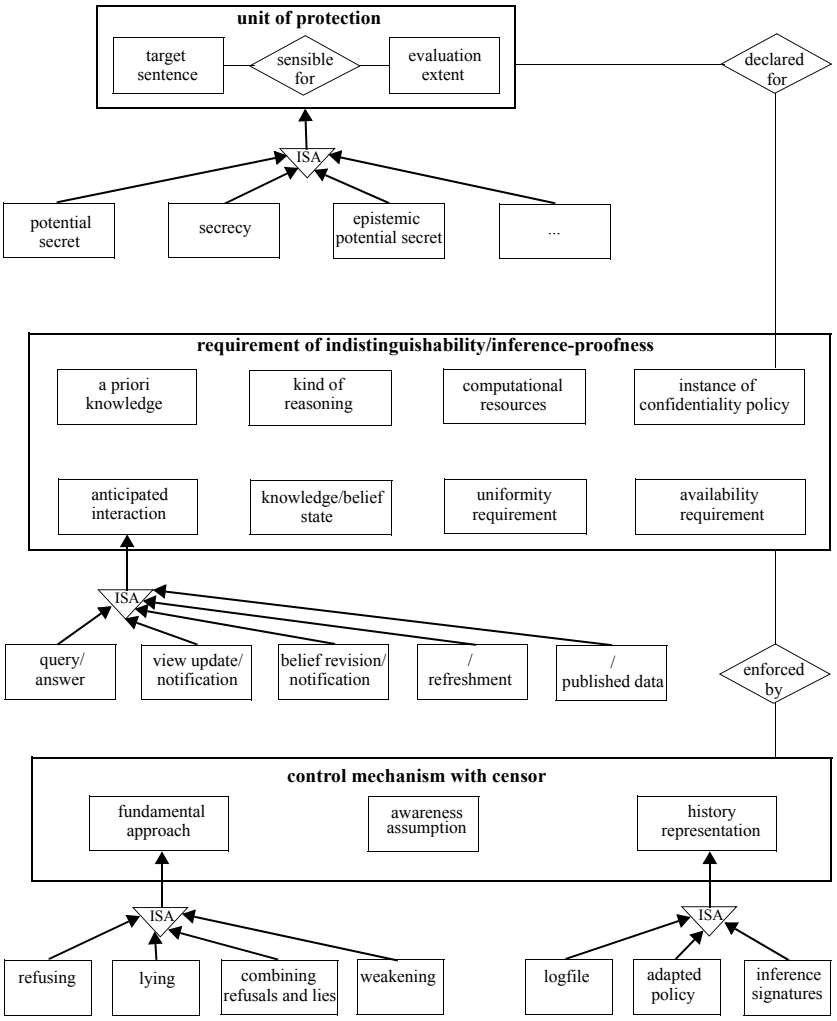

Fig. 2. ER model of features of a CIE confidentiality policy

2 Syntax: What Could Be Kept Confidential?

The vague question "What could be kept confidential?" gets the brief general answer "Parts of the data provider's knowledge or belief, respectively – ", which could be extended by " – as far as these parts can be expressed by means of a vocabulary agreed by the data provider and the client". This general answer, however, might have many different specializations.

2.1 Units of Self-protection

In fact, the particularity of inference-usability confinement by means of CIE is that the control mechanism is solely located at the *data provider*, and the *client* is not at all hindered in his activities. In particular, the client can observe the (fictitious) real world by himself, communicate with other agents, and send any messages to the provider. Consequently, the provider is only able to self-protect features that strictly belong to herself, namely her own knowledge or belief, respectively, and to control her own activities, whether spontaneous or reactive on messages received. Moreover, only those parts of the provider's epistemic state are meaningful units of protection that are within the scope of the communication agreement between the provider and the client.

More precisely, we are modeling the situation sketched above as follows. The data provider maintains a *knowledge/belief* base. All *interactions* refer to this knowledge/belief base according to a communication agreement, which in particular describes the underlying logic with its basic vocabulary, including constant symbols, predicate symbols and logical operators. In principle, any syntactically admissible sentence saying "The provider knows/believes the sentence ... " might be a meaningful *target* of protection. Moreover, depending on the semantics of the underlying logic, any evaluation results of relating such a sentence to the actual knowledge/belief of the provider might form a possible *extent* of protection. Thus a *unit* of protection is the combination of a target with an extent. Notably, two syntactically different but semantically equivalent targets will lead to equally handled protection units, and in this sense information will be protected independently of its syntactic representation.

2.2 Epistemic Potential Secrets

This approach is formally elaborated in various ways. Most prominently, assuming an incomplete knowledge base under proof-theoretic semantics, a target sentence can be evaluated to any of the three different values *true*, *false* and *undefined*. For example, the value *undefined* for some sentence intuitively says "The provider does not know that the sentence is *true* and the provider does not know that the sentence is *false*". Any nonempty subset of the possible evaluation values can be selected as an extent that then requires that the client is not allowed to infer the validity of any of its elements. To achieve the intended goals, *modal logic* (of knowledge/belief) is employed, and a protection unit is represented as a so-called *epistemic potential secret* [25, 26, 42].

If the provider's knowledge base is complete, an evaluation returns either *true* or *false* and, accordingly, proof-theoretic semantics of the underlying logic can be replaced by model-theoretic semantics. So a security officer can decide to either protect just one value – conveniently only the value *true* allowing the negation operator for a target – or both values. In the former case we speak about "potential secrets", and in the latter case about "secrecies" [1, 6, 7, 39]. If acceptable by the application, to enable an optimized control in some work potential secrets have been restricted to target sentences expressing a so-called "fact" in an

ONF (BCNF + UniqueKey) schema [11, 13], or existentially quantified atomic sentences [13–15, 17, 18, 36].

Extending CIE to an information system that is based on an advanced logic might require us to suitably adapt the notion of a unit of protection. For example, Wiese [44] treats *possibilistic knowledge bases* employing a "standard possibilistic logic" and then considers necessity-valued units of the form (φ, α), where φ is a target sentence and $\alpha \in (0..1]$ a necessity *weight*. In another example, Biskup/Tadros [23] treat *defeasible belief bases* employing "ordinal conditional functions", and a further extension of this work could consider *ranks* of sentences.

Modal logics and other advanced logics are often constructed by only extending an underlying *propositional logic*, though many applications would suggest to extend a fragment of first-order logic. Unfortunately, however, in many cases handling an underlying first-order logic would lead to considerable logic-technical difficulties; as an alternative, under some conditions an application formalized in a suitable fragment of first-order logic could be first propositionalized and then handled by an extension of propositional logic, see [24] for an example.

2.3 Decidability and Finiteness

In an algorithmic treatment, in general each individual unit of protection will be inspected whether or not it could be violated by a message sent to the client. There are two basic problems: *decidability* of violation regarding one unit, and *finite representation* of infinitely many units and then overall *termination* of control. The former problem will be considered in Section 4. The simplest solution to the latter problem is just to require that the policy instance contains only finitely many units of protection.

However, an application might require us to specify an *infinite set*. For example, dealing with a relational database having a relation scheme $\mathtt{Patient}(\mathtt{Name}, \mathtt{Disease})$, we might want to protect targets of the kind that some patient `Bob` suffers from *whatever* disease. Supposing a vocabulary with an unlimited type extension $\{\mathtt{d}_1, \mathtt{d}_2, \mathtt{d}_3, \mathtt{d}_4, \ldots\}$ for the attribute `Disease`, we would then get the following infinite set of targets: $\{\mathtt{Patient}(\mathtt{Bob}, \mathtt{d}_1), \mathtt{Patient}(\mathtt{Bob}, \mathtt{d}_2), \mathtt{Patient}(\mathtt{Bob}, \mathtt{d}_3), \mathtt{Patient}(\mathtt{Bob}, \mathtt{d}_4), \ldots\}$.

Essentially, there are two approaches to come up with a finite representation. First, we can always slightly strengthen the policy by replacing such an infinite homogeneous enumeration by a single existentially quantified sentence, which in our example is $(\exists \mathtt{x})\mathtt{Patient}(\mathtt{Bob}, \mathtt{x})$: if the confidentiality of this sentence is successfully enforced, then all sentences in the infinite enumeration are kept confidential as well. Second, we can abbreviate an infinite homogeneous enumeration by using an open formula with free variables that range over the pertinent type extensions, in our example resulting in the open formula $\mathtt{Patient}(\mathtt{Bob}, \mathtt{x})$. We then have to assure that the inspection procedure always terminates after checking a finite subset of the range. Such an assurance has been proved explicitly for the optimized censors presented in [14, 18].

3 Semantics: What Does Confidentiality Mean?

Having discussed *what* could be kept confidential, a more challenging question arises: "What does confidentiality of a unit of protection actually mean?" A brief answer is "The client *cannot exclude* a (correct) evaluation of the target *outside* the extent".

Slightly expanded, this answer requires the following: "The client will never *be enabled to exclude* the possibility that in the provider's knowledge/belief the target of the unit is (correctly) evaluated *differently* from each value in the extent of the unit". Speaking in terms of a belief of the client: "The client will *never believe* that the provider's (correct) knowledge/belief regarding the target is *within* the extent". In yet another words: "The client *cannot distinguish* the actual (correct) situation of the data provider from a fictitious alternative situation in which the target would be evaluated *outside* the extent". These already complicated answers are still in need to be complemented by further specifications:

- the postulated *a priori knowledge* and *computational capabilities* that the client could employ for attempting to violate confidentiality;
- the *anticipated interactions* between the data provider and the client, together with the corresponding messages sent and received;
- the *states in time* of the provider's knowledge/belief that are considered to be in need of protection;
- the *uniformity* of protection, i.e., whether all units in the policy are protected together or, alternatively, each unit in the policy is treated independently of the others;
- the requirement on *availability* to avoid a trivial understanding of confidentiality that would essentially deny to communicate data to the client at all.

3.1 Indistinguishability – From the Client's Point of View

The answers sketched above to the question about semantics are specializations of a very general approach: the client should obtain an *inference-proof view* on the actual situation in which the provider's knowledge/belief is hidden from the client unless parts of the knowledge/belief have been revealed by interactions between the provider and the client. Basically, the various specializations result from different notions of a "situation". The general approach can be outlined as follows:

For all "situations" held to be possible by the client,
considering a unit of protection in the policy instance,
there exists an alternative possible "situation" such that
– from the point of view of the client –
(1) the actual and the alternative situation are indistinguishable,
in particular all (re)actions of the provider visible to that client are the same,
(2) but in the alternative situation
the evaluation of the target of the unit is not in the extent of the unit.

As exemplarily proved in [20], such a notion of inference-proofness is in the spirit of extensive related research on "possibilistic secrecy", as concisely summarized in [34].

3.2 A Priori Knowledge and Capabilities – As Postulated

Clearly, the notion of a "situation" also includes internal properties and behaviors of the client, in particular his *a priori knowledge* of the provider's knowledge/belief and his kind of *reasoning* about the provider's knowledge/belief and the corresponding computational capabilities. Though the provider has agreed to cooperate with the client in principle, the provider nevertheless sees the client as a potential attacker of her confidentiality interests. Seen as an attacker, the client presumably will not communicate his own internal properties and behaviors to the provider, and thus the provider has to *postulate* the client's internal properties and behaviors.

Then the effectiveness of the notion of inference-proofness as given above crucially depends on a fundamental *simulation assumption*: The provider postulates the client's actual internal properties and behaviors correctly, or at least approximates them appropriately. Consequently, in charge of constructing the control mechanism for the provider, a security engineer has to design and implement a suitable *simulation* of the client at the site of the provider, and this simulation has to be effective without any assistance of the client.

Regarding the *a priori knowledge* of the client, the provider has to make a "good guess". Regarding the *kind of reasoning* of the client, the provider has to postulate the client's perspective of how the provider forms her hidden knowledge/belief. So, in principle, we would have to consider a mutual recursive model: First, of course, the provider is fully aware of how she herself forms her knowledge/belief; the client has his private perspective on this formation; in turn the provider has to simulate how the client perceives the formation of the provider's knowledge/belief; and the client might argue about this simulation; ... . In practice, however, we somehow have to terminate the recursion smoothly: this is exactly the purpose of the simulation assumption.

In most of our work, for the sake of simplicity, the provider and the client are supposed to employ the same classical logic, whether propositional logic, a feasible fragment of first-order logic or, more generically, a logic just described by some natural properties. In these cases, the simulation assumption holds by default. In our recent work on multiagent systems [22, 23], however, the provider is modeled to employ some fixed reasoning with defeasible conclusions under belief revisions, and the client is postulated to use a skeptical entailment operator based on a specific class of defeasible conclusions: the provider then simulates exactly this usage. In a first version [23], that class is semantically specified by considering all ordinal conditional functions; in a forthcoming version [22], that class can be selected more generally by an axiomatization satisfying dedicated "allowance properties".

Finally, not only the client's kind of reasoning is important but his actual *capabilities* in terms of computational resources as well. To be on the safe side,

as usual in security engineering, so far we always postulate the client to be computationally omnipotent, i.e., not restricted to a feasible class of algorithms. Clearly, more advanced considerations taking care of the achievements of complexity theory would be highly worthwhile.

3.3 Anticipated Interactions

The notion of a "situation" also refers to *sequences of interactions* between the data provider and the client, actually executed in the past and potentially performed in the future. In our work, interactions include query answering, update processing, belief revision and data publishing.

A client may send a message containing a *query* request to the provider, who will return a message with an *answer*. The query might be *closed*, essentially asking for the evaluation of a communicated query sentence by the provider's knowledge/belief. A query might also be *open*, essentially asking for the evaluations of all sentences that result from some substitution of free variables by constant symbols in a communicated query formula. Clearly, closed queries can be seen as a special case of open queries. However, there is a crucial feature of open queries that does not apply for closed queries: given an infinite set of constant symbols, infinitely many sentences arise from considering all possible substitutions, and so we need assumptions and conventions to guarantee that answers are finitely representable.

Employing a well-known approach for relational databases, we treat open queries based on the following assumptions and conventions [9]: (1) The finitely many atoms represented by the stored relation instances are evaluated positively (regarded as *true*). (2) Seeing the database as providing complete information, by means of a closed-world assumption all remaining atoms are assumed to be evaluated negatively (regarded as *false*). (3) Each query sentence must be domain-independent and thus safe, to guarantee that only finitely many substituted sentences are evaluated positively. (4) All remaining substituted sentences are assumed to be evaluated negatively. (5) The infinite set of remaining sentences is finitely represented by an appropriate *completeness sentence* that can be expressed in first-order logic. Most crucially, in general the control mechanism necessarily has to inspect not only the positive part of an answer but also the pertinent completeness sentence. A first step towards dealing with open queries to an incomplete information system applies a suitable propositionalization of the first-order constructs under the restrictive assumption of having only finitely many constant symbols [24].

Regarding updates [12, 21], a client might send a *view update* request on which the data provider reacts by returning a notification about the actions internally taken to reflect the update in her knowledge/belief. If the knowledge/belief is actually modified, other clients – if there are any – should get informed by suitable *refreshments* of previously released data. Furthermore, the data provider herself can initiate an update of her knowledge/belief, and then send refreshments to the clients. Basically, there are two kinds of threats regarding confidentiality.

First, a notification about a successful or rejected update contains answers to *implicit queries*, namely whether or not the new information was already present before and whether or not the *semantic constraints* are satisfied after a requested modification. Clearly, these answers have to be controlled similarly as answers to explicit queries to prevent a violation of confidentiality. Moreover, it makes a difference whether not only *single updates* but also whole *transactions* are handled: in the latter case, new and already available information could be mixed, and semantic constraints are enforced by finally either aborting or committing all steps, while in between some constraints might temporarily not be satisfied. Second, in principle the client could gain hidden information by comparing and relating refreshed information with aged information, which again has to be controlled.

In advanced cases, when the provider forms her knowledge/belief by means of some defeasible and thus non-monotonic reasoning, the data provider might also accept requests of *belief revision* [22, 23], and then notifies the client whether or not the communicated revision is in conflict with her unchangeable belief and thus must be rejected (and in current work we also treat conflict resolution and corresponding notifications about *how* accepted revisions are performed).

An interaction might also occur in a degenerated form, namely as *data publishing* autonomously initiated by the data provider [16, 27, 28]. Afterwards, a client can use the published data at his own discretion. Basically, data publishing can be seen as responding to a query about the pertinent part of the data. So far, however, we have studied only the case that just this single query is treated, leaving *iterated releases* and their potential threats for future work, see, e.g., [41, 45, 46].

More generally, in principle it would be worthwhile to consider arbitrarily mixed sequences formed by all kinds of interactions. Extending the remark above, however, so far we only treated some special cases, and each case results in a different notion of possible "situations".

3.4 Knowledge/Belief States under Protection

An important aspect of a "situation" is the knowledge/belief of the data provider. If no updates occur and only queries are considered, the initial knowledge/belief state remains invariant over the time. Accordingly, this fixed state is referred to in the definition of indistinguishability given in Subsection 3.1. However, if there are updates and revisions, then a "situation" comprises a sequence of knowledge/belief states, started with the initial knowledge/belief state, exhibiting the stepwise produced intermediate states, and ending with the current knowledge/belief. In that case, we distinguish when the evaluation of a target of a unit of protection is required to be outside the extent: either *temporarily* only in the current (last) state or *continuously* in all states of the sequence [12, 21].

3.5 Uniformity of Indistinguishability

The indistinguishability property outlined in Subsection 3.1 somehow vaguely refers to "considering a unit of protection in the policy instance". Clearly, it makes a great difference how this wording is formalized precisely: either "for each unit of protection there exists an alternative situation (possibly different for each unit) with the wanted properties regarding that particular unit" or "there exists an alternative situation with the wanted properties uniformly for all units of protection".

The semantic difference has also been captured in a syntactic way: either the policy instance contains several units with unrelated targets or, w.l.o.g., the policy instance consists of just one disjunctive target formed from all sentences identified to be in need of protection. In fact, all algorithms following the lying approach as discussed in Section 4.1 need to protect the disjunction of all targets anyway, see, e.g., [7, 9, 26–28, 30].

3.6 Availability Requirements

In general, we have to balance interests in confidentiality on the one hand with legitimate interests in availability of information needed on the other hand. While an interest in confidentiality is basically represented by the data provider, an interest in availability is mainly but not exclusively held by the client. Requests for availability can be successively expressed in at least three layers:

- *Explicitly and extensionally*, by listing a set of sentences that definitely should never be distorted, see, e.g., [10, 26–28, 40, 42, 43]: we then have to ensure that this listing is not in conflict with the instance of the confidentiality policy, either by removing or weakening at least one of the sentences being in conflict, or by ranking conflicting requirements.
- *Implicitly and intensionally*, by applying some kind of *meta-rule* that the data provider may distort data to be communicated to the client *only if* this is strictly necessary for preserving confidentiality within the given setting, as followed in all work surveyed in this report: mostly, the meaning of "strictly necessary" is captured by an heuristic that otherwise some straightforward violation of confidentiality could be exhibited.
- *Measured*, by requiring a *minimum distance* between functionally correctly executed interactions and controlled interactions, see [27, 28, 43]: we then have to find a convincing notion of distance and will face additional challenges to master the resulting optimization problem.

4 Algorithms: How to Decide on a Single Request?

Once both a policy instance – declaring *what* should be kept confidential – and the wanted policy semantics – describing what the client should *invariantly not believe* now and in future – have been specified, for each single interaction the data provider has to decide the question "What should be said to the client

if the correct reaction was harmful?". There are two fundamental approaches which can also be combined: "*Refuse* to show a meaningful reaction!" or "Tell a plausible *lie*!". Furthermore, a more informative variant of the refusal approach suggests "Suitably *weaken* the reaction!".

More generally, to comply with the semantics of the policy instance, the data provider needs a control mechanism that stepwise *censors* each individual request issued by the client and generates an immediate reaction returned to him. The main challenge is to determine individual reactions that taken together do not violate the policy, and will not contribute to do so in future! Clearly, the construction of an appropriate censor will strongly depend on the policy semantics including the accompanying specifications chosen for an application. Besides the intended semantics the following issues are most important:

- The selection of a fundamental approach has a crucial impact: *refusing* requires the data provider to carefully consider options of the client to profit from meta-inferences; *lying* needs care to avoid running into a hopeless state of affairs in the future; *combining* refusals and lies might be helpful to avoid both meta-inferences and hopeless situations; *weakening* attempts to only return harmless information that is true but, as far as possible, more expressive than just a refusal notification.
- Though we always postulate that the client is fully aware of both the overall approach of inference-usability confinement and the specific policy semantics in operation, we might distinguish whether or not the client is assumed to be *aware* of the actual policy instance.
- Facing the worst case that the client might remember the complete history of all previous interactions, the data provider has to *represent the history*, too, whether directly by *keeping a logfile* or indirectly by *adapting the policy instance* or by employing some mixed form.
- There might be options to select *optimized* censors for different classes of policy instances. In this case, we could take advantage of relating the static *expression complexity* of a policy instance and possibly further items to the dynamic *runtime complexity* of a censor.

4.1 Refusing, Lying, Combining Refusals and Lies, and Weakening

The *refusal approach* appears to be a most natural way to prevent a violation of confidentiality by an interaction initiated by the client: the data provider just denies to return a meaningful reaction to the client but instead notifies the client about the detection of a potential threat [1, 4, 6, 7, 9, 11, 23, 26, 36, 39]. However, to avoid enabling the client to perform meta-inferences based on the refusal notification, the provider has to employ so-called additional refusals in cases that seem to be non-critical at first glance. In a nutshell, from the point of view of the client, a refusal should always have an explanation in terms of a "harmless situation" that is indistinguishable from a potential "harmful situation".

The *lying approach* could be seen controversial by ethical reasons or practical interests in availability: whether interactively or by own initiative, the data

provider delivers data that might not faithfully reflect the evaluation(s) in her actual knowledge/belief state(s), clearly without revealing an actual deviation from the correct information [6, 7, 9, 12, 26–28, 30, 44]. However, in general the data provider has to treat more information as sensitive than originally seen to be units of protection, in order to avoid running into a hopeless state of affairs in the future. Such a bad event would arise if the client has acquired knowledge about a positive evaluation of a disjunction of target sentences all of which are protected; the client could then query all theses sentences, would get lies returned and could then detect an inconsistency in the provider's reactions.

The *combined approach* aims at avoiding the technical shortcomings of uniform refusals and uniform lies, namely blocking of meta-inferences and protecting disjunctions of sensitive data, respectively, [8, 9, 26]. In an interaction with the client, if the correct reaction is harmful, the data provider first checks whether a lie would be harmless and actually returns it only in that case; otherwise, if both correct and lied information would lead to a violation of confidentiality, a refusal notification is returned. In this way, meta-inferences are avoided, since the client cannot discriminate the correct case and the lied case, and disjunctions need no special consideration, since lies are explicitly checked.

Instead of just refusing, equivalently only returning a tautology, the *weakening approach* aims at returning only true information – in particular to avoid the ethical concerns and availability problems of lies – but as much as possible without violating confidentiality [14, 16, 19]. Accordingly, the provider generates return data such that the corresponding information can be seen as an entailment of the correct information, in an extreme case just a tautology as a trivial entailment. This approach has only preliminarily been explored for CIE and will be inspected more closely in forthcoming work. A crucial problem is the client's kind of awareness of weakenings, and his related options to profit from meta-inferences.

For example, originally designed as a kind of refusal, the treatment of open queries in relational databases [14] is based on switching from closed world evaluations to open world evaluations. Then a missing sentence in the answer returned to the client has the following weakened meaning: either the provider's knowledge/belief evaluates the sentence to *false* or the provider's control mechanism has removed it for the sake of confidentiality. This two-sided disjunction can be seen as a weakening of the first side.

For another example, treating XML documents as incomplete data (in an open world), inference-proof publishing of XML documents removes directly or indirectly sensitive nodes from an XML tree but then needs to modify the underlying DTD before showing it to the client [16]. Here, weakening occurs in two forms: nodes not shown to the client may be "either not present or discarded", and the new DTD basically expresses less requirements than the original DTD.

Fragmentations of relational data into a projection visible to a client and a projection kept hidden by the provider have been shown to be inference-proof in the sense of our notion of indistinguishability under some reasonable but restrictive assumptions [19]. Here, a (hidden) value of an attribute not shown in the

visible projection can be interpreted as being replaced by its type as declared in the schema. And the type of a value can be seen as a weakening of that value, expressible again as a disjunction or by means of existential quantification, depending on the cardinality of the type extension. In ongoing work, we are further exploring weakenings for CIE by means of disjunctions inspired by approaches proposed for achieving k-anonymity and l-diversity of published relational data, see, e.g., [31, 33, 37]. Here a "generalized value" can also be seen as a disjunction of the original values that contribute to the generalization.

4.2 Awareness of Policy Instance

In most cases, we follow the conservative assumption that the policy instance is *known* by the client, i.e., the client is aware about the dedicated sentences to be kept secret to him.

A more relaxed assumption is that the policy instance is *unknown* to the client [1, 7, 8, 14, 39, 42], such that the client's uncertainty about the provider's actual knowledge/belief also extends to the policy instance. Accordingly, in reasoning about "situations", the client has to consider more possibilities. The increased variety then offers additional options for the control mechanism to hide the secret part of the actual situation. In fact, under the assumption of an unknown policy instance, the necessary control tends to be essentially less restrictive. For example, for a complete information system and potential secrets as units of protection, the control basically has to consider only those parts of a policy instance that are *true* in the provider's knowledge/belief, whereas the remaining *false* parts can be neglected. Moreover, the optimized censor for open queries presented in [14] would fail to preserve confidentiality for a known policy instance in some cases.

4.3 Representing the History

A straightforward solution to the data provider's task of representing the history of preceding interactions with a client is keeping a *logfile* of all messages and a *repository* containing the time-stamped sequence of knowledge/belief states. Then, in principle, the provider could always reconstruct all her observations, simulations and actions over the time. Often, more efficient solutions can be designed.

For example, if only *queries* about *complete* information can occur [7] it suffices that the provider retains only the sentences returned to the client as answers. These sentences together with the sentences describing the postulated a priori knowledge of the client then appropriately simulate the client's view on the provider's knowledge/belief. Moreover, the set of all these sentences might contain redundancies which can be removed. In particular, since tautologies are redundant, there is no need to retain refusal notifications. Notably, however, answers to queries about possibly *incomplete* information have to be converted into sentences in modal logic of knowledge [26].

If *updates* are also anticipated and therefore *refreshments* have to be sent to clients [12], it does *not* suffice to store a non-redundant *set* of sentences that is equivalent to the set of sentences returned as answers. Rather we need to remember the order in time of their disclosure to the client to generate inference-proof refreshments, since refreshments have to be produced by re-evaluating the corresponding sentences in the order of their original processing. Moreover, if the client himself may request *view updates*, the last point in time of processing such a request defines a special event, which is treated similarly as an initialization.

Besides explicitly logging all messages or at least the modal sentences representing the provider's reactions, the provider can also treat the history in some kind of a preprocessed form. Roughly outlined, inspecting a possible reaction, the provider's censor has to relate the following two items: the simulated, dynamically evolving view of the client on her knowledge/belief tentatively augmented with sentences representing the reaction on the one hand, and the units of protection declared in the instance of the confidentiality policy beforehand or suitably derived sentences on the other hand. The pertinent relationship is given by an *entailment* operator that is appropriate for the underlying logic. Accordingly, a main task of the censor is algorithmic *theorem proving* for that logic.

More precisely, the censor has to find formal proofs of harmful entailments, pertinent to the selected fundamental approach and further parameters. Sentences in the simulated view of the client are treated as assumptions of an entailment, and a sentence under protection as a conclusion. If such a formal proof can be successfully completed, indicating that the entailment actually holds, then the inspected reaction is forbidden to be shown to the client. Otherwise, if all attempts to find a proof of a harmful entailment fail, the reaction can be shown to the client safely and the simulated view could be incremented accordingly.

However, following an alternative way referred to as *policy adaptation* [4], the provider might aim at profiting from previous unsuccessful proof attempts: the missing parts of an attempted but failed proof are seen as indicating *proof obligations* still to be inspected for future interactions. Under some assumptions, the censor might gather all relevant proof obligations and suitably convert them into an adapted policy instance and then discard the sentences used as assumptions. More generally, instead of simply employing a logfile, a history could be represented by at least two dynamically updated components that suitably complement each other: (1) a logfile containing sentences of the simulated view of the client as far as they are still needed in future for the censor's inspections and (2) an adapted instance of the confidentiality policy representing the currently postulated proof obligations of the client.

In the straightforward solution, the logfile is always incremented by all sentences returned to the client, and the policy instance remains fixed; for the special case investigated in [4], the logfile is not needed any more. In another special case, controlling queries to relational data constrained by template dependencies [15], a mixed approach is followed: in a static *preprocessing phase* all possible formal proofs of harmful entailments are generated, and subsequently, in the dynamic *control phase*, these precomputed proofs are employed like intrusion signatures,

stepwise marking the actual progress the client could achieve for these proofs based on the sentences returned to him.

Clearly, for the sake of high efficiency, we would like to get rid of theorem proving at all: neither keeping a logfile, nor adapting the policy instance, only comparing an inspected sentence with sentences specified in units of protection by means of simple string operations, preferably even expressible in standard SQL in the case of a relational database.

4.4 Optimized Censors

The efficiency goal sketched above can actually be achieved for some restrictive cases of relational data. Our considerations focus on the refusal approach, assuming that the client is aware of the policy instance. Basically, we explore four cases, where the first case is the standard one without any optimization. In all cases, we have to find a sufficient and "reasonably necessary" control condition.

Case 1. As long as decidability is achieved, any a priori knowledge, policy instance and closed (yes/no-)queries (of the relational calculus) are admitted (also extended to safe open queries with closed-world semantics). While maintaining and employing a logfile that represents the a priori knowledge and the answers to previous queries, we have to ensure that adding neither the correct answer to the current query nor the negation of the correct answer will be harmful; additional refusals for harmful negations of correct answers guarantee that an observed refusal notification cannot be traced back to its actual cause by exploiting the system awareness [6, 9].

Case 2. The a priori knowledge may only comprise a schema declaration with functional dependencies that lead to Object normal form, i.e., Boyce-Codd normal form with a unique minimal key. Confidentiality policies are restricted to select-project sentences – having an atomic body with constant symbols (for selection) and existentially quantified variables (for projection) – of a special kind referring to "facts", and queries are restricted to arbitrary select-project sentences. Without maintaining a logfile, it suffices to ensure that the query sentence does not "cover" any policy element [11].

Case 3. The a priori knowledge is restricted to only comprise a schema declaration with functional dependencies (without further restrictions). Policy instances are restricted to (arbitrary) select-project sentences, whereas queries must be closed select-queries – being an atomic formula with constant symbols but without existentially quantified variables. Without maintaining a logfile, it suffices to ensure that the query sentence does not "cover" any policy element [18].

Case 4. The a priori knowledge is restricted to only comprise a schema declaration with functional dependencies and full join dependencies (without any further restrictions). Confidentiality policies and queries are restricted to (arbitrary) select-project sentences. We have to ensure two conditions: (1) The query sentence does not "cover" any policy element. (2) Previous positive answers together with a positive answer to the current query do not "instantiate" any template dependency that is both implied by the schema dependencies and "covering" an element of the confidentiality policy [14, 15].

5 Pragmatics: How to Fit an Application?

Being well-trained in the variety of options for policy instances, policy semantics and algorithmic censors and charged with the responsibility for an application, a security officer still faces the task "How to fit the application most appropriately?". Our general advice is twofold: "Follow tool-supported business rules, and rely on automatic optimizations". Clearly, suitable tools and optimizations have to be constructed and included in the overall control mechanism beforehand.

5.1 Tools for Policy Administration

In an ongoing project we are implementing a prototype for CIE, based on the underlying Oracle database management system and the theorem prover Prover9, see [5]. Basically, the prototype can be used in two modes: in the *administration mode*, a security officer can register a user as a client, in particular declaring all parameters for the pertinent confidentiality requirements imposed on the user; in the *user mode*, the interactions with each registered client are automatically controlled according to the security officer's declarations.

We are currently designing comprehensive tools for assisting the security officer, and integrating them into the administration mode of the prototype. Figure 3 outlines the overall workflow of these tools. The figure mainly shows the configuration activities of the *security officer* as a human individual followed by the computations of the data provider's *server* which finally result in various items maintained by the *database*. The security officer bases his work on the accomplishments of the *database administrator* who declares the application schema including the semantic constraints and creates the initial instance of the data provider's knowledge/belief.

The security officer starts his own activities by letting the server create a database for the items needed for the control. Additionally, to prepare for open queries under potential lies, the security officer may specify dictionaries for "convincingly inventing" constant symbols that are not occurring in the actual instance, see [9, 28] for further details. Then the security officer possibly iteratively performs four subactivities: setting up published or common knowledge to be included into the a priori knowledge of every client; defining generic user roles for facilitating the administration of many users; modeling a particular user including his postulated a priori knowledge beyond the published and common knowledge and imposing a dedicated policy instance on him; tentatively selecting a censor for later on controlling the interactions of a particular user.

All these subactivities are coordinated by the server, often in an interactive way asking the security officer for approvals or alternatives. In particular, the server automatically checks whether the gathered a priori knowledge and the policy instance are in conflict, and whether all further preconditions of a considered censor are satisfied.

Finally, for each particular user, the security officer can commit all his decisions and thereby transfer the actual control activities to the server, which then will communicate with the respective client in user mode.

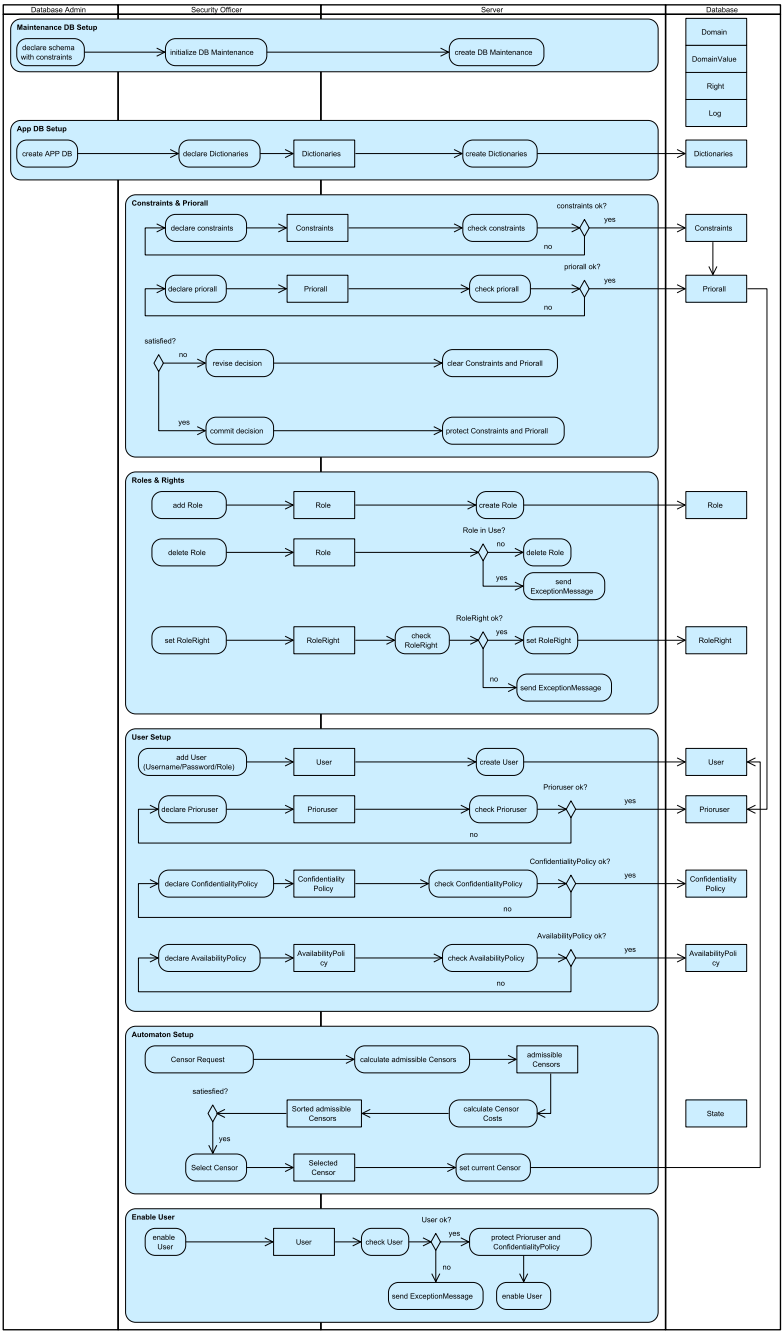

Fig. 3. Workflow for configuration of the CIE prototype including policy administration

5.2 Automatic Optimization

In general, the security officer will select and finally confirm all parameters pertinent for a client, and then the control mechanism will automatically inspect and confine all interactions with that client accordingly. Nevertheless, over the time it might turn out that the declarations are not fully adequate and could be improved in some way, in particular a more suitable censor could be chosen. We are currently designing a semi-automatic optimization component, and integrating it into the user mode of the prototype, foreseeing interventions of the security officer as far as needed. Figure 4 outlines the overall workflow of the optimization component.

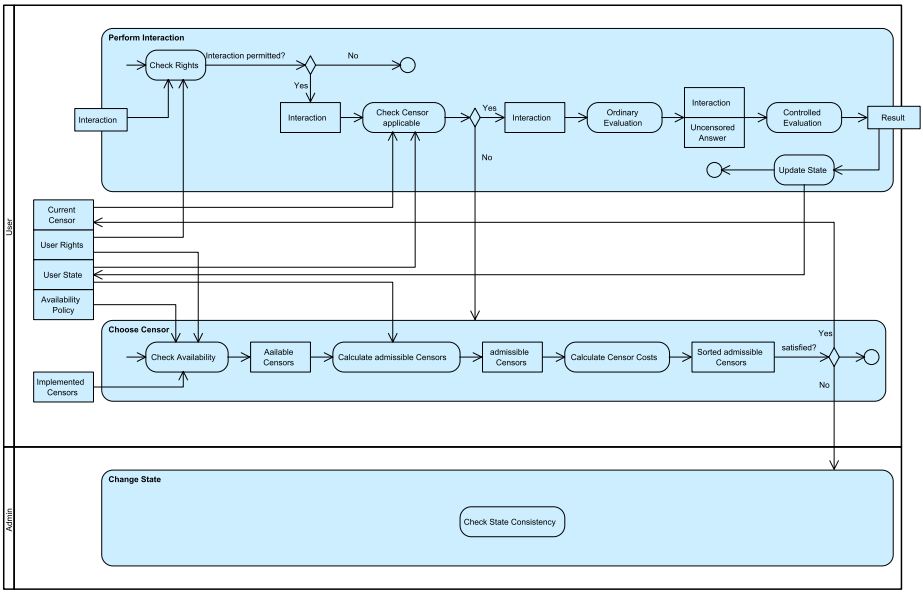

Fig. 4. Workflow of an automaton for optimized censor selection

6 Conclusions and Challenges

In our work on CIE we favor a control mechanism that is based on a logic-oriented confidentiality policy: essentially, a security officer specifies what information should be kept confidential to a particular client, and then the data provider's control mechanism automatically enforces the declaration over the time when interacting with that client and communicating data to him. As a first insight conveyed by this report, we emphasize that the security officer has to carefully consider many subtle details and further aspects beyond just the

"what": semantics and algorithms also need to be configured suitably. Accordingly, effective administration tools appear to be mandatory. So far, we only studied a relatively small set of parameter selections taken from a still widely unexplored and *huge range of options*.

The overall approach of CIE aims at achieving confidentiality regarding a client solely by confining the information content of messages communicated by the data provider to that client, who is supposed to be an intelligent agent. As a second insight, we emphasize that the provider's control mechanism has to appropriately model and simulate that client. Clearly, modeling and simulation are based on postulates about the client, who is suspected to (maliciously) exploit principally permitted information sharing for goals not covered by an agreement with the data provider. Thus the client cannot be expected to be cooperative in finding a simulation that matches the potential or even the actual behavior. Accordingly, achieving confidentiality crucially depends on the quality of those *necessarily uncertain postulates*.

Moreover, since in the CIE approach the data provider will not at all intervene at the site of the client, the provider can only *protect her own knowledge/belief* about the "mini-world" of an application. In contrast, she cannot prevent the client from learning from other sources or even colluding with other agents. Accordingly, all such possibilities have to be captured by good postulates, for example regarding other sources by declaring specific a priori knowledge, and regarding collusion by treating a suspicious group of clients like a single client.

Finally, even if the challenges mentioned above can be mastered, *computational complexity* and in particular *scalability* remain urgent problems. Indeed, as a third insight, we emphasize that the unavoidable subtask of simulating an intelligent agent in general requires us to employ *sophisticated knowledge engineering* including *costly theorem proving*.

Obviously, inference control – and CIE in particular – are dealing with only one aspect of a more general security problem, namely to bind the sharing of a specific piece of information to a dedicated *purpose* determined by the subject of the information. Accordingly, many further control mechanisms together with pertinent policies have been designed. We only exemplarily mention two lines of research. Based on a commonly accepted infrastructure, e.g., as supplied by trusted computing, obligation and *usage control* employs a tamper-proof security component at each site to block unwanted activities on data tagged with a policy statement, see, e.g., [35, 38]. Taking advantage of the large amount of data and the corresponding information available in the Semantic Web, *reactive policies* allow to guide and control the behavior of an agent in the web, see, e.g., [29].

Acknowledgments. I would like to sincerely thank Marcel Preuß and Cornelia Tadros for continuous and fruitful cooperation on many aspects of CIE. Moreover, I am indebted to the current members of the "Task Force CIE" formed by the master students Martin Bring, Katharina Diekmann, Dirk Schalge, and Jaouad Zarouali: they are successfully elaborating and implementing the ideas on tools and optimization to be included into a CIE prototype.

References

1. Biskup, J.: For unknown secrecies refusal is better than lying. Data Knowl. Eng. 33(1), 1–23 (2000)
2. Biskup, J.: Usability Confinement of Server Reactions: Maintaining Inference-Proof Client Views by Controlled Interaction Execution. In: Kikuchi, S., Sachdeva, S., Bhalla, S. (eds.) DNIS 2010. LNCS, vol. 5999, pp. 80–106. Springer, Heidelberg (2010)
3. Biskup, J.: Inference control. In: van Tilborg, H.C.A., Jajodia, S. (eds.) Encyclopedia of Cryptography and Security, 2nd edn., pp. 600–605. Springer, Heidelberg (2011)
4. Biskup, J.: Dynamic policy adaption for inference control of queries to a propositional information system. Journal of Computer Security 20, 509–546 (2012)
5. Biskup, J.: Inference-usability confinement by maintaining inference-proof views of an information system. International Journal of Computational Science and Engineering 7(1), 17–37 (2012)
6. Biskup, J., Bonatti, P.A.: Lying versus refusal for known potential secrets. Data Knowl. Eng. 38(2), 199–222 (2001)
7. Biskup, J., Bonatti, P.A.: Controlled query evaluation for enforcing confidentiality in complete information systems. Int. J. Inf. Sec. 3(1), 14–27 (2004)
8. 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)
9. 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)
10. Biskup, J., Burgard, D.M., Weibert, T., Wiese, L.: Inference Control in Logic Databases as a Constraint Satisfaction Problem. In: McDaniel, P., Gupta, S.K. (eds.) ICISS 2007. LNCS, vol. 4812, pp. 128–142. Springer, Heidelberg (2007)
11. 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)
12. Biskup, J., Gogolin, C., Seiler, J., Weibert, T.: Inference-proof view update transactions with forwarded refreshments. Journal of Computer Security 19, 487–529 (2011)
13. Biskup, J., Hartmann, S., Link, S., Lochner, J.-H.: Chasing after secrets in relational databases. In: Foundations of Data Management, AMW 2010. CEUR Workshop Proceedings, vol. 619, pp. 13.1–13.12 (2010)
14. 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)
15. 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)
16. Biskup, J., Li, L.: On inference-proof view processing of XML documents. IEEE Transactions on Dependable and Secure Computing, 1–20 (2012), doi:10.1109/TDSC.2012.86
17. Biskup, J., Lochner, J.-H.: Enforcing Confidentiality in Relational Databases by Reducing Inference Control to Access Control. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 407–422. Springer, Heidelberg (2007)

18. Biskup, J., Lochner, J.-H., Sonntag, S.: Optimization of the Controlled Evaluation of Closed Relational Queries. In: Gritzalis, D., Lopez, J. (eds.) SEC 2009. IFIP AICT, vol. 297, pp. 214–225. Springer, Heidelberg (2009)
19. 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)
20. Biskup, J., Tadros, C.: Policy-based secrecy in the Runs & Systems Framework and controlled query evaluation. In: International Workshop on Security (Short Papers), IWSEC 2010, pp. 60–77. Information Processing Society of Japan (2010)
21. 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)
22. Biskup, J., Tadros, C.: Preserving confidentiality while reacting on iterated queries and belief revisions (2012) (submitted)
23. 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)
24. 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)
25. Biskup, J., Weibert, T.: Confidentiality Policies for Controlled Query Evaluation. In: Barker, S., Ahn, G.-J. (eds.) Data and Applications Security 2007. LNCS, vol. 4602, pp. 1–13. Springer, Heidelberg (2007)
26. Biskup, J., Weibert, T.: Keeping secrets in incomplete databases. Int. J. Inf. Sec. 7(3), 199–217 (2008)
27. Biskup, J., Wiese, L.: Preprocessing for controlled query evaluation with availability policy. Journal of Computer Security 16(4), 477–494 (2008)
28. Biskup, J., Wiese, L.: A sound and complete model-generation procedure for consistent and confidentiality-preserving databases. Theoretical Computer Science 412, 4044–4072 (2011)
29. Bonatti, P.A., Kärger, P., Olmedilla, D.: Reactive Policies for the Semantic Web. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 76–90. Springer, Heidelberg (2010)
30. Bonatti, P.A., Kraus, S., Subrahmanian, V.S.: Foundations of secure deductive databases. IEEE Trans. Knowl. Data Eng. 7(3), 406–422 (1995)
31. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Samarati, P.: K-anonymity. In: Secure Data Management in Decentralized Systems. Advances in Information Security, vol. 33, pp. 323–353. Springer (2007)
32. Farkas, C., Jajodia, S.: The inference problem: A survey. SIGKDD Explorations 4(2), 6–11 (2002)
33. 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) (2010)
34. Halpern, J.Y., O'Neill, K.R.: Secrecy in multiagent systems. ACM Trans. Inf. Syst. Secur., 12(1), 5.1–5.47 (2008)
35. Kelbert, F., Pretschner, A.: Towards a policy enforcement infrastructure for distributed usage control. In: Atluri, V., Vaidya, J., Kern, A., Kantarcioglu, M. (eds.) Access Control Models and Technologies, SACMAT 2012, pp. 119–122. ACM (2012)

36. Lochner, J.-H.: An Effective and Efficient Inference Control System for Relational Database Queries. PhD thesis, Technische Universität Dortmund (2011), http://hdl.handle.net/2003/27625
37. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: *l*-diversity: Privacy beyond *k*-anonymity. TKDD, 1(1) (2007)
38. Pretschner, A., Hilty, M., Basin, D.A., Schaefer, C., Walter, T.: Mechanisms for usage control. In: Information, Computer and Communications Security, ASIACCS 2008, pp. 240–244. ACM (2008)
39. 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)
40. Tadros, C., Wiese, L.: Using SAT-Solvers to Compute Inference-Proof Database Instances. In: Garcia-Alfaro, J., Navarro-Arribas, G., Cuppens-Boulahia, N., Roudier, Y. (eds.) DPM 2009. LNCS, vol. 5939, pp. 65–77. Springer, Heidelberg (2010)
41. Wang, K., Fung, B.C.M.: Anonymizing sequential releases. In: Eliassi-Rad, T., Ungar, L.H., Craven, M., Gunopulos, D. (eds.) Knowledge Discovery and Data Mining, KDD 2006, pp. 414–423. ACM (2006)
42. Weibert, T.: A Framework for Inference Control in Incomplete Logic Databases. PhD thesis, Technische Universität Dortmund (2008), http://hdl.handle.net/2003/25116
43. Wiese, L.: Preprocessing for Controlled Query Evaluation in Complete First-Order Databases. PhD thesis, Technische Universität Dortmund (2009), http://hdl.handle.net/2003/26383
44. Wiese, L.: Keeping Secrets in Possibilistic Knowledge Bases with Necessity-Valued Privacy Policies. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) IPMU 2010. LNCS, vol. 6178, pp. 655–664. Springer, Heidelberg (2010)
45. Xiao, X., Tao, Y.: M-invariance: towards privacy preserving re-publication of dynamic datasets. In: Chan, C.Y., Ooi, B.C., Zhou, A. (eds.) Management of Data, SIGMOD 2007, pp. 689–700. ACM (2007)
46. Yao, C., Wang, X.S., Jajodia, S.: Checking for k-anonymity violation by views. In: Böhm, K., Jensen, C.S., Haas, L.M., Kersten, M.L., Larson, P.-Å., Ooi, B.C. (eds.) Very Large Data Bases, VLDB 2005, pp. 910–921. ACM (2005)