Policy-Based Data Downgrading: Toward A Semantic Framework and Automated Tools to Balance Need-To-Protect and Need-To-Share Policies

Grit Denker, Ashish Gehani, Minyoung Kim
Computer Science Laboratory
SRI International
Menlo Park, CA 94025 USA
Email: Firstname.Lastname@sri.com

David Hanz
Engineering and Systems Division
SRI International
Menlo Park, CA 94025 USA
Email: David.Hanz@sri.com

***Abstract*—We describe a new paradigm for articulating need-to-protect and need-to-share policies that shows promise for enabling automated derivation of the downgrading rulesets needed to comply with these policies in systems that share data. This new paradigm is based on fine-grained semantic policy specifications in terms of context, content, Purpose, and Anti-purpose that are expressed in a machine-understandable language. Our approach is based on an existing reasoning capability that can handle simple downgrading cases. Extensions to handle more complex cases are discussed. Although not yet a complete, turnkey solution to the overall data sharing and privacy problem, we posit that our approach provides an auspicious research vector for future work towards achieving that goal.**

***Keywords*-privacy, data sharing, formal policy, need-to-protect, need-to-share, semantic models, downgrader, analyzer, generator**

I. Introduction

Ubiquitous and distributed computing applications must share data to deliver the quality of services they are designed to achieve — but whenever data is shared, privacy risks are taken. Take, for example, the U.S. Department of Homeland Security: To effectively protect the public from terrorism, agencies must collaborate; but for the most part, these agencies do not have joint formal privacy agreements. Whether in social networking, health care, or other applications, every American citizen wants to trust that data pertaining to him will be available to those who have a legitimate need, and that this data will not be misused. *Downgrading* is the process of transforming data so that information is protected according to privacy policies, but still contains the information needed to perform computations or provide required levels of service. Today's computing and communication systems employ tailor-made downgrading solutions that are not reusable, and the expert knowledge that went into the design and implementation of these downgraders is lost in the details of hardware configurations or software code. There is a need for well-founded models and analysis tools to formulate and enforce both privacy (i.e., need-to-protect) policies and need-to-share policies.

We propose a formal, semantic framework for modeling and analyzing need-to-protect and need-to-share policies, to enable articulating and reasoning about these often contradictory policies. The proposed solution relates data to the activities it enables, and identifies these activities as either harmful (i.e., corresponding to need-to-protect policies) or beneficial (i.e., corresponding to need-to-share policies). This fits the diverse needs of the wide range of privacy stakeholders. We term this new approach *Policy-Based Data Downgrading* (PBD), where policy refers not only to need-to-share and need-to-protect policies, but also to various other context and data content-specific information. Expert knowledge of the rationale for downgrading data in certain circumstances is captured in models; and thus is reusable in contexts characterized by similar data or policies.

Usability of the PBD is increased through automated tools that balance need-to-share and need-to-protect policies and provide downgrading solutions by selecting and combining downgrader operations. Appropriate downgrading is accomplished by determining a set of arithmetic or logical operations to be performed on the data, altering it in such a way that the data simultaneously enables the beneficial activities and precludes the harmful activities. Based on an existing Analyzer that determines system interoperability using semantic markup, we will develop a novel PBD Analyzer that uses constraint-solving mechanisms to determine the right combination of downgrading operations. Future plans also include development of a novel Downgrader Specification Generator (DSG) for cases in which downgrading operations are missing. Together, the PBD Analyzer and DSG constitute an important step toward a full circle of automated tools for data downgrading. The tools will take the burden off users and make automated enforcement of privacy policies in systems feasible. The proposed solution is comprehensive: it will work with static and streaming data, and with changing downgrading needs.

II. Related Work and What Is Still Missing

The PBD paradigm assumes a *producer* transmitting data to a *downgrader* that sanitizes the data before forwarding it to the *data consumer*. This is similar to the *Privacy-Preserving Data Publishing* (PPP) framework, wherein data originates from an *individual record owner* and is collected

by a *publisher* that releases a compendium to the public or specific *recipients* [1]. Mechanisms to downgrade information have been under development for several decades. Adams [2] described three classes for PPP: (1) restricting the number of queries that can be made, (2) perturbing the data before executing the query; or (3) running the query on unmodified data, but then perturbing the output. In the PBD, the recipient does not make an explicit query, so the first category is not relevant. However, techniques that operate on the data can be useful for PBD. These techniques include *generalization*, which coarsens, abstracts, or collects multiple data into equivalence classes; *suppression*, which removes records from the sanitized output; *swapping*, which interchanges attributes from different records; *randomization*, which adds random noise to perturb the data; and *multi-views*, which apply different variants of the previous techniques to provide diverse sanitizations [3]. While generalization, suppression, and randomization can be directly applied in the PBD setting swapping may not be, since it would require buffering in the downgrader and may cause unintended semantic changes as the data is streamed to a recipient, and multi-views is superseded by the PBD's more flexible mechanism for customizing the output.

The PBD data consumer and PPP recipient are the only adversaries assumed. In particular, the PBD downgrader and PPP publisher are trusted to operate correctly. We do not consider the untrusted case where the sanitization is effected with a cryptographic protocol [4]. There are, however, numerous other differences between the PBD and PPP settings, as described below.

PPP schemes are designed with the assumption that all the data has been collected by the publisher and, thus, sanitization can operate in an *offline* mode, preprocessing all the data prior to answering any queries. PBD schemes may be used in settings where the data is being streamed through the downgrader, necessitating algorithms that adapt *online*. In particular, since individual data records may need to be downgraded en route to a specific recipient, removal of *identifiers* is not generally feasible. This precludes guarantees such as differential privacy [5] in the PBD setting. In addition to the fact that the data set is encountered online in PBD, the relevant data set may be in flux. In contrast, in PPP schemes data sets are *static*, allowing the publisher to compute statistical characteristics and make sanitization choices ensuring that particular properties hold in the absence of external auxiliary information.

PPP frameworks are inherently *centralized* as they assume that the downgrader has access to the entire database. In contrast, PBD schemes are *decentralized*, operating on data as it flows through information channels in a system.

In addition, PPP sanitization procedures have a fixed *Anti-purpose* (i.e. need-to-protect policy), which is to prevent *re-identification*, although the specific *quasi-identifier* attributes being protected and the notion of identification may differ. Similarly, if the inability to distinguish a record from $k-1$ others is deemed to prevent re-identification, then k-anonymity [6] suffices; whereas if the statistical similarity of sensitive attributes is a concern, then other assurances like l-diversity [7] would be needed. In the case of PBD, the Anti-purpose could be more complex [8], such as predicating the inferences the recipient will be able to draw on knowledge of the data content and the context in which the data is being used.

PBD, like PPP, does not aim to provide perfect sanitization (i.e., the prevention of an attacker from learning any more information than would have been possible without access to the protected collection of data). In contrast to PPP where the *partition* of sensitive and non-sensitive attributes is static [1], PBD constraints define whether a particular attribute is sensitive as a function of the context, content, purpose, and policy restrictions. Thus, another difference in the frameworks is that PBD can flexibly target goals for specific pieces of data, while PPP schemes make statistical guarantees about collections of published data.

There exists some work on semantic models for policy-driven information exchange. The KAoS policy language is used to formalize privacy and information sharing policies [9]. However, while this approach uses a similar modeling framework as that envisioned for the PBD, the KAoS solution is not as general as our approach, and considers only very simple downgrading operations that do not require the sophisticated constraint reasoning to determine an appropriate combination of downgrading operations and parameters needed to address a broad range of privacy and data sharing policies.

Our PBD is also similar to Risk-Adaptable Access Control (RAdAC) [10] in its goal to tradeoff need-to-share policies with the risk of releasing information. RAdAC approaches do not apply downgrading; they weigh risk with need-to-share and decide whether or not to release data in its original form. Our work could complement this approach in providing the means to automatically balance between these policies.

To summarize, despite signs of progress toward a downgrading solution, two central issues remain: (1) Need-to-share policies are typically defined in vague terms, if at all. This makes it difficult to correctly implement the policies, especially since there is a companion need-to-protect imperative. (2) No general technical solutions exist to automate the derivation of downgrading rules from privacy policies. Without an automated derivation, the concept of forming ad hoc, agile data-sharing systems-of-systems becomes hopeless.

We believe that these issues have a common thread that can be resolved only by obtaining precise answers to two *Why* questions: (1) *Why do I need to share?* and (2) *Why do I need to protect?* Although one approach has been to label data with security classifications such as Top Secret or

Secret, the problem with this approach is that the *rationale* for assigning a particular security label to a category of data is not captured. By *rationale*, we mean the precise details about the harmful activities that a potential adversary could perform, if given access to the data. We offer the conjecture that without a precise understanding of the details of that rationale, it is impossible to assess whether a "sanitized" version of the information would preclude those specific harmful activities. We also make the observation that the security labels are noun-like entities, whereas the harmful actions that constitute the rationale must be expressed using verbs (the grammatical construct that denotes activity).

III. Limitations of the Proposed Approach

The scope of the overall data privacy and sharing problem space is somewhat staggering in its breadth, so we restrict our approach to a subset of the overall problem space. The following limitations apply to our approach:

1) The most significant limitation of our approach is that it deals only with fixed-format, structured-data messages or files. Our ontological modeling relies on this assumption.
2) The terms *High Side* and *Low Side* are customarily used to denote the two domains between which a data-sharing solution is intended to adjudicate data flow. While the usual connotation is that the Low Side is a proper subset of the High Side, we will use this terminology without implying such restricted interpretation. Each side may hold information that it is not allowed to share with the other. However, our approach deals with data flow from High-to-Low.
3) We assume that low side recipients are specifically identified (including a special label for public access).
4) We consider the following factors in determining downgrading: content of current information instance only [memoryless decision], current instance plus some previous instances [n-state memory decision], memoryless or n-state memory decisions plus static or dynamic preconditions [i.e., state information that remains constant or may change during a specific time period].
5) We assume that it is not necessary to conceal the fact that downgrading has been applied.
6) We consider both cases where the data being downgraded is a singleton or an instance of a sequence of related information objects.

IV. Historic Downgrading Example Scenario: GPS Selective Availability

The Global Positioning System (GPS) was conceived in 1973 by the U.S. Department of Defense to provide navigation and location information to both military personnel and civilian users. The initial deployment in 1978 consisted of four satellites that were visible in a limited region (four being the minimum number needed for a receiver to calculate its position in 3-D space). By 1995, twenty-four satellites were in orbit, providing complete worldwide geographic coverage.

Concern that an enemy might use the GPS information for a harmful purpose was addressed by designing the system to provide two levels of service, one with high accuracy for the military and another with lower accuracy for civilian use [11]. The dual service was constructed by structuring the GPS transmission as two signals, L1 and L2, transmitted at 1575.42 MHz and 1227.6 MHz, respectively. A deliberate downgrading is performed on L1 to degrade the quality of the information provided on that signal. This downgrading process, *Selective Availability* (SA), was implemented by adding a pseudorandom error to the L1 signal's time component, thus perturbing the quality of the position, velocity, and time information that could be derived from the signal. The information derived from L1 is the *Standard Positioning Service* (SPS), and allows the horizontal surface position (corresponding to x, y coordinates in a local tangent plane) to be calculated within a 100 meter accuracy, the local vertical position (altitude z coordinate) to be calculated within a 156 meter accuracy, and the time to be calculated within 340 nanoseconds of the reference atomic clock. This uncertainty in the location and time values was deemed sufficient to prevent the GPS data from being used for adversarial purposes, such as precision targeting and guidance of weapons.

In contrast, the L2 signal is not perturbed, but is encrypted to prevent unauthorized persons from utilizing it. The absence of SA on L2 data and the fact that L2 is transmitted on a different frequency than L1, allows a comparison between the L1 and L2 data to be used to factor out errors resulting from signal traversal through the atmosphere. The resulting *Precise Positioning Service* (PPS) service has lower levels of uncertainty than the SPS: 22 meters for horizontal position, 28 meters for local vertical/altitude, and 200 nanoseconds for time.

Although the U.S. Government discontinued the application of SA to the L1 signal in 2000 (by setting the magnitude of the added noise bias to zero), we argue that the original scheme represents a data-sharing example where downgrading rules were derived from the need-to-protect and the need-to-share policies:

- **Need-to-protect policy:** Precise targeting and guidance for a long-range weapon must be denied to an adversary.
- **Need-so-share policy:** Support for long-range area navigation must be possible using a degraded-accuracy version of the GPS information.

In this GPS example, the underlying functions to be performed for privacy policy and sharing policy are fun-

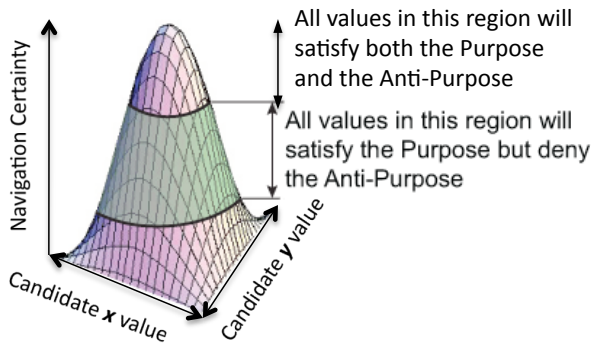

Figure 1. GPS downgrading example: Downgrading within the marked region enables *Purpose* while preventing *Anti-purpose*.

damentally the same[1] — it is only the accuracy of the information required that changes. Figure 1 illustrates the region of x, y coordinates of GPS where downgrading satisfies the goals of *Purpose* (enabling appropriate levels of GPS information) and the *Anti-purpose* (precluding access to precise GPS information by unauthorized users). The z coordinate represents a navigation quality metric. The Purpose requires any value above the lower boundary, which effectively specifies the minimum accuracy needed. The Anti-purpose rules out providing any value that is too accurate (i.e., a value that falls above the upper boundary). The region between the boundaries contains values of x and y that are valid candidate values for a downgrader to produce that will permit the Purpose to be achieved while also denying the harmful activities described by the Anti-purpose.

V. A New Paradigm for Need-To-Protect and Need-To-Share Policies

We propose a new paradigm for articulating need-to-protect and need-to-share policies, wherein data is (1) related to activities, and (2) these activities are identified as either harmful or beneficial. By doing so, we make the rationale for the assigned security classification explicit. This paradigm provides a framework for expressing the answers to the *Why* questions mentioned earlier:

> Q: Why do I need to protect this information?
> A: Because, if released, this information could enable the recipient to perform <*detailed, formal description of* **harmful activity**>.
> Q: Why do I need to share (a downgraded version of) this information?
> A: Because I need to enable the receiver to perform <*detailed, formal description of* **beneficial activity**>.

Reduced to its simplest form, the problem of determining an appropriate downgrading ruleset becomes one of finding a set of arithmetic/logical operations to be performed on the data, altering the data in such a way that it simultaneously:

- Enables the beneficial activity — *we call this the* ***Purpose***
- Precludes the possibility of the harmful activity — *we call this the* ***Anti-purpose***

Of course, it must be recognized that, for any given instance, there may exist no such simultaneous solution.

We believe these notions of Purpose/Anti-purpose are profound and have actually been underlying most previous cross-domain ruleset implementations — but usually without recognition and without being captured or expressed in a formal representation.

VI. Technical Approach: Modeling and Analysis Framework for Downgraders

The Purpose/Anti-purpose construct is one semantic model that contained in the first component of our framework. Other semantic models pertain to the content and context of the data, resources that provide or receive data, and formal downgrading specifications (see Figure 2). The second main component of the envisioned framework comprises the Analyzer and Downgrader Specification Generator tools. The Analyzer uses semantic models stored in various knowledge bases (KBs) to assess *whether and how downgrading can be achieved for a given Purpose/Anti-purpose goal*, and applies KB facts to determine the appropriateness of existing downgrader specifications to achieve this goal. The Analyzer will return with one of four possible answers:

1) *Yes*: No downgrading is necessary because the selected resources can already perform the Purpose activities, and the Anti-purpose activities are not possible.
2) *Yes + Downgrader Specification*: Downgrading is necessary and the Analyzer selected the appropriate formal downgrader specifications from the KB after successfully validating that after downgrading the Purpose activities are still possible and the Anti-purpose activities cannot be performed.
3) *No + Downgrader Specification Constraints*: Downgrading is necessary, but none of the existing downgrader specifications in the KB satisfy the Purpose and Anti-purpose constraints. However, the Analyzer can return constraints resulting from the reasoning process and use them as input for a downgrader specification generation phase.
4) *No*: There are two cases to distinguish: (a) either Purpose and Anti-purpose are inconsistent, which means that there is no solution; or (b) the selected resources do not have the capabilities to perform the Purpose activities (e.g., satellites do not generate accurate enough GPS data to be used for navigation). In the latter case, the Analyzer can inform the user about the missing capabilities of the selected resources. The

[1]This is not a necessary condition for our need-to-protect/need-to-share concept, but it makes the essence of the concept easier to visualize.

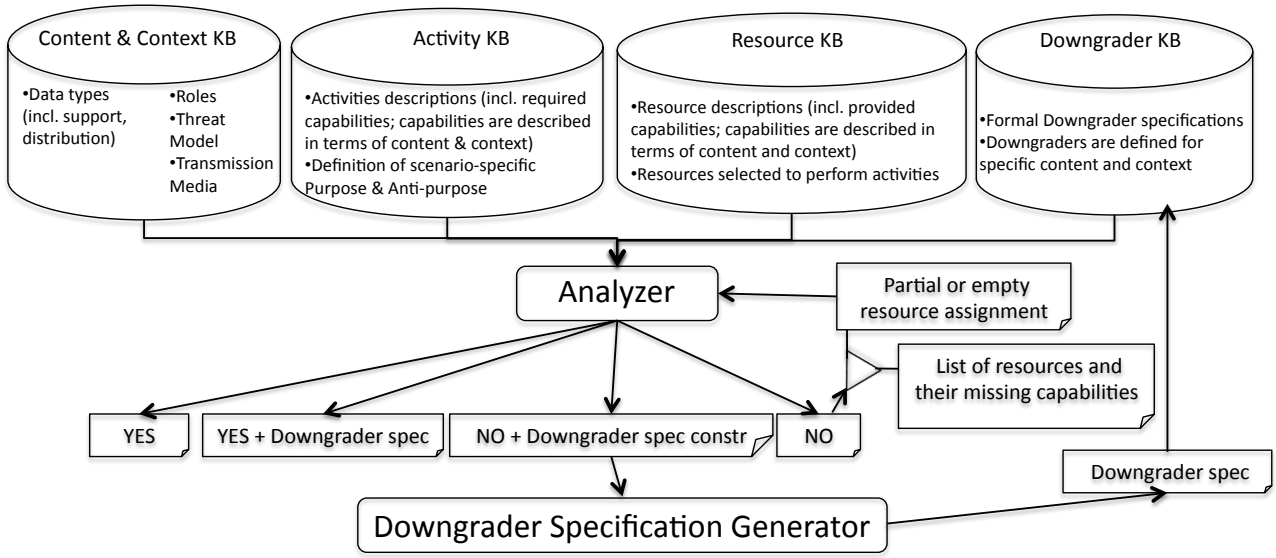

Figure 2. Overview of semantic models and downgrading tools.

Analyzer can also be run again with partial or no resource assignments and will iterate through all the resources to see whether any exist that can satisfy the Purpose/Anti-purpose constraints.

Our PBD approach is based on the ontological framework developed under a U.S. Department of Defense sponsored project called the *Open Net-centric Interoperability Standards for Training and Testing* (ONISTT) [12], [13], [14] which provides a semantically rich means of describing machine-to-machine interactions in terms of a hierarchical construct known as the *Task*. The Tasks (referred to as activities in this paper) are described in terms of the capabilities needed to perform that activity (i.e., the data types with their qualitative or quantitative characteristics needed to perform that activity). While the ONISTT framework does not target privacy applications, many of the concepts developed there can be reused as a basis for a modeling and analysis framework for downgraders. The ontological models of ONISTT contain more than 60 ontologies, and more than 75% of those ontologies are general-purpose models that are independent of the training and testing domain for which they were developed.

In the ONISTT project, we have been very successful in using OWL [15] and SWRL [16] to model the task ontology as well as domain-specific knowledge such as location data. These approaches will suffice for much of our PBD research, and where they do not, we will use mathematical logic notations to yield the required expressiveness[2].

[2]Machine-readable syntax to exchange policies is not our foremost matter. Rather, we want to focus on providing the Analyzer capabilities needed to solve the downgrading problems automatically.

The ONISTT task ontology provides a good starting point for a generic activity ontology. ONISTT's resource ontology can be reused in the privacy context for entities that generate, send, or receive data. For example, in the GPS use case, satellites producing GPS data are resources, and so are the military personnel and general public users who consume GPS data to perform navigation or targeting activities.

New models must also be developed. In particular, formal specifications of downgraders are a core ingredient of our semantic framework. Downgrader specifications are different from informal downgrading rulesets that commonly result from BOGSATs (a Bunch Of Guys Sitting Around a Table). Downgrader specifications also different from downgrader appliances and their technical specifications. Rather, downgrader specifications are formally specified rulesets that define the data types on which the rules can be applied and that unambiguously specify the effect of applying these rulesets to the data. They can be used as a basis on which we design and implement downgrader appliances, and to certify them against these specifications (though this is not within the scope of this project). Downgrading specifications formally capture the essence of informal downgrading rulesets while abstracting from implementation details of downgrader appliances. Downgrading specifications are used by the Analyzer to determine whether any specifications exist that can be used to achieve Purpose and prevent Anti-purpose.

VII. Toward a Solution for Automated, Model-Based Downgrading

We will briefly illustrate our approach with the help of the GPS SA example. For clarity, we will use abbreviated versions of the full semantic models described elsewhere

[8]. GPS data is semantically modeled by a concept *GeocentricPosition* for which properties x, y, and z are length measurements that have two properties defined: *measurand* and *uncertainty*. For our example, the *uncertainty* is of interest, and it is defined as a tuple containing a *magnitude* value and a *unit*. Thus, a GPS element that indicates 20 meters uncertainty for position x can be noted as *x.uncertainty.magnitude = 20* and *x.uncertainty.unit = meter*. In the following formulas, we assume all units to be meters and do not further mention them.

For the GPS SA example we define two activities:

Navigate(?GPSProvider,?GPSConsumer) and
Target(?GPSProvider,?GPSConsumer).

The terms *?GPSProvider* and *?GPSConsumer* are role parameters for the activities that can be assigned to resources. To navigate, the *?GPSProvider* must provide GPS data with accuracy of at least 100 meters; and to target, GPS data must be accurate to at least 20 meters. These constraints are formally modeled in our framework as

NavigateConstraint:=
x.uncertainty.magnitude $\leq$ *100* $\wedge$ *y.uncertainty.magnitude* $\leq$ *100* $\wedge$ *z.uncertainty.magnitude* $\leq$ *100*

TargetConstraint: =
x.uncertainty.magnitude $\leq$ *20* $\wedge$ *y.uncertainty.magnitude* $\leq$ *20* $\wedge$ *z.uncertainty.magnitude* $\leq$ *20.*

What constitutes Purpose or Anti-purpose depends on the specific situation. Thus, a human chooses some activities to be Purpose while other activities are chosen to be Anti-purpose. In our framework, special predicates *Purpose, Antipurpose*, and *Goal* are defined as *Goal* $\Leftrightarrow$ *Purpose* $\wedge$ $\neg$ *Antipurpose*. The specifics of these predicates are defined for each example.

For our example, using the individuals *GeneralPublic* and *MilitaryPersonnel* for the *?GPSConsumer* role of the navigation and targeting activities, and leaving the role parameter *?GPSProvider* unassigned, we define two goals:

Goal1 $\Leftrightarrow$
(Purpose $\Leftarrow$ *Navigate(?GPSProvider,GeneralPublic))* $\wedge$
$\neg$*(Antipurpose* $\Leftarrow$ *Target(?GPSProvider,GeneralPublic))*

Goal2 $\Leftrightarrow$
Purpose $\Leftarrow$ *(Target(?GPSProvider,MilitaryPersonnel)* $\wedge$
Navigate(?GPSProvider,MilitaryPersonnel)) $\wedge$ $\neg$
(Antipurpose $\Leftrightarrow$ *false)*

For the case that GPS data is sent to the general public, we want to enable navigation with it, but prevent it from being used for targeting. For the case that the receiver of the data is military personnel, we want to be able to do both, navigation and targeting. The question is *Which resources provide GPS data with the appropriate accuracies for Goal1 and Goal2?*. We assume two satellite resources *Sat1* and *Sat2*, both with the capability to provide GPS: the former with uncertainty of at most 10 meters and the latter with uncertainty of at most 30 meters. Our Analyzer can validate that satellite *Sat1* satisfies *Goal2* and that the GPS data provided by *Sat2* satisfy *Goal1*. If none of the resources were assigned, our Analyzer could determine resource assignments that satisfy the goal.

In the case that downgrading is necessary (e.g., downgrade GPS provided by *Sat1* to satisfy *Goal1*) the Analyzer can use existing downgrading specifications. We assume that downgrader specifications contain at least the following information. *Downgrader(?in_type, ?in_unc, ?param, ?out_type, ?out_unc)* where *Downgrader* is the name of the downgrader specification. *?in_type* is the input data type for this downgrader specification, which usually refers to one of the standard data types (e.g., GPS or integer). *?in_unc* defines the uncertainty of the input data type. (This includes the error distribution and the support of the distribution. The support of the distribution is the smallest interval or set whose complement has probability zero. It may be understood as the points or elements that are actual members of the distribution. Thus *?in_unc* is usually defined as a tuple *(support, distribution)*. Examples of error distribution are standard normal (i.e., Gaussian) distribution or uniform distribution. The term *([120,130],uniform)* is an example of a *(support,distribution)* tuple where the uniformly distributed values range from 120 to 130.)

Returning to the downgrader specification above, *?param* describes parameters of the downgrader specification (for example, a bit-dropping downgrading specification has *?bits* parameter to determine the number of bits that it drops). *?out_type* is the specification of the data type that will be returned by the downgrader specification. *?out_unc* is the error distribution and its support for the resulting data type after downgrading.

These formal downgrader specifications make up the Downgrader KB (see Figure 2). The Analyzer iterates through the specifications to find ones that will ensure that goals are satisfied. Thus, the Analyzer tries to prove that the goal still holds after one substitutes the original input data type/distribution with the data type/distribution produced after applying the downgrader operations, as below:
Downgrader(?in_type,?in_unc,?param,?out_type,?out_unc)
$\Rightarrow$ *Goal[?in_type,?in_unc* $\leftarrow$ *?out_type,?out_unc]*.

For this example, let us assume the x coordinate of the reported GPS data from *Sat1* to be 125 with support [120,130] and uniform distribution. This corresponds to at most 10 meter inaccuracy, which is the same as that defined for *Sat1*. We aim to downgrade this data to have at least 20 meter inaccuracy. To ensure that we have the same characteristics for the data before and after downgrading, we need to maintain the original mean. Thus, after downgrading, the mean should still be 125, but with support [115,135] and uniform distribution. How does the Analyzer come to

the same conclusion after a formal reasoning argument? While we intuitively argued what should be the output after downgrading on a specific GPS value, our approach to automated downgrading is more general. The base for providing such automated tools are detailed semantic specifications of downgraders and their effect on data types. Thus, in our example, we know that the data type we are trying to downgrade is integer and we know that its distribution is uniform. As such, the Analyzer attempts to find a solution for *DG(integer, [L_i, R_i], uniform, ?param, integer, [L_o, R_o], uniform)* $\Rightarrow$ *Goal1*. We start by determining values for the output support interval [L_o, R_o]. Since we know the input support interval [L_i, R_i], we can compute the output support interval, by substituting pairs of input-output variables into the following linear equation:

$$y = f(x) = a * x + b \tag{1}$$

where constants a and b would be uniquely determined through substitution of two different values. While a linear function is sufficient to solve the mapping from [L_i, R_i] to [L_o, R_o], when L_i and R_i are known, it is also a simple representation. In particular, if it is known that a linear equation was used for solving the downgrading challenge, an attacker could invert the downgrading and compute the original support interval. In our framework we will study downgrading operations of various complexity and relate them to privacy requirements. We note that in some cases there is a temporal aspect to how rigorous the downgrading operation must be in order to withstand attacks (e.g., in our GPS scenario, detailed location data might not be sensitive after several minutes or hours).

Substitution of L_i and L_o first and then R_i and R_o leads to the following:

$$L_o = a * L_i + b \text{ and } R_o = a * R_i + b \tag{2}$$

Subtracting the two equations from each other leads to $(L_o - R_o) = (L_i - R_i) * a$, and substituting a with $\frac{(L_o - R_o)}{(L_i - R_i)}$ in equation (2) results in $L_o = \frac{(L_o - R_o)}{(L_i - R_i)} * L_i + b$. Therefore, the equation $y = a * x + b$ can be solved as

$$y = f(x) = \frac{(L_o - R_o)}{(L_i - R_i)} * x + L_o - \frac{(L_o - R_o)}{(L_i - R_i)} * L_i. \tag{3}$$

Thus, to realize the necessary downgrading operation, a parameterized multiplication operation and shift operation is required. For multiplication, it takes x as its input and $\frac{(L_o - R_o)}{(L_i - R_i)}$ as a parameter for a in (1) to generate $a * x$ value. Subsequently, the shift operation takes the $a * x$ value as its input and $L_o - \frac{(L_o - R_o)}{(L_i - R_i)} * L_i$ as a parameter for b in (1) to generate the final result $y = f(x)$. Substituting [120,130] in (3) for $[L_i, R_i]$ yields $y = 2 * x - 125$, which will transpose [120,130] to [115,135] as informally argued earlier.

If the Downgrader KB contains specifications for *Mult(?param)* and *Shift(?param)* downgrading operations, the Analyzer can find these specifications and determine that they are appropriate in this case. If no such downgrader specifications are available, the Analyzer can use the result of the reasoning (3) to generate such downgrader specifications.

Note that in the case where the input value is exact (i.e., the support of the input values is [L_i, L_i]) we use the following alternative to get an input support interval with two different boundaries: First we drop the least significant bits using a *BitDrop* operator to yield the intermediate support [L'_i, R'_i] where $L'_i < R'_i$ and $(R'_i - L'_i)$ is smaller than $(R_o - L_o)$. *BitDrop(?bits)* is a parameterized downgrading operator that drops *?bits* least significant bits from the input. In the case described, the Analyzer would determine *?bits* to be the bit that is less than the most significant digit of $(R_o - L_o)$.

While the illustrated example posed only a simple downgrading challenge, in the general case the Analyzer would have to perform more complex reasoning about downgrader specifications and their combinations.

Other Examples: MTTC and TCAS

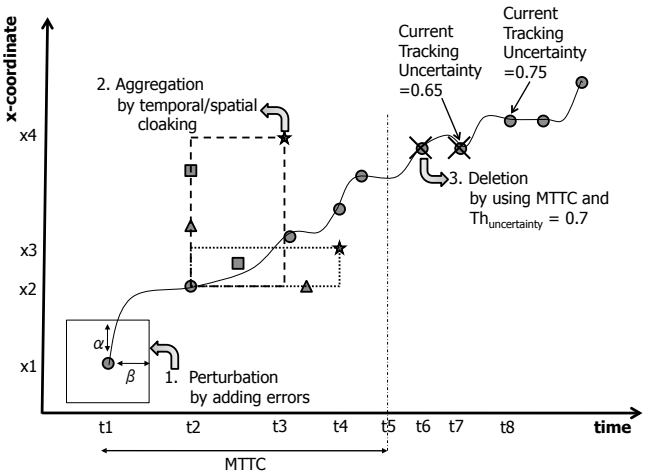

Figure 3. Sample downgrades: 1. Pertubation, 2. Aggregation, 3. Deletion.

To illustrate the applicability of our approach, we present two more examples. First, consider communication of GPS data from a control station to private vehicles in an urban freeway setting. The Purpose is to assist the driver in route planning and navigation of her vehicle considering current traffic situations, while the Anti-purpose is the capability to track individual vehicles. Figure 3 shows the time elapsed on the x-axis and one dimension of the location on the y-axis. Each symbol (circle, triangle, rectangle) represents different individuals. The most intuitive way of downgrading is perturbing the values of either the location coordinates (e.g., $x_1 \pm$ 1 km) or timestamp (e.g., $t_1 \pm$ 1 min) by introducing numeric errors α and β as illustrated in Figure 3. However, perturbation without careful consideration of privacy guarantees may not support our goal (e.g., perturbing

sufficiently many bits to avoid revealing critical information may also preclude the Purpose). Thus, a more structured way of downgrading is achieved by specifying an anonymity parameter k which is the minimum size of the set of indistinguishable objects. The k-anonymity policy decreases the accuracy of the GPS data by choosing a sufficiently large area (spatial cloaking) or time interval (temporal cloaking) so that enough other objects exist within the area or time interval to satisfy the anonymity constraint. In Figure 3, the GPS data has 3-anonymity in the temporal domain (i.e., $k_{time} = 3$ at t_2) and 2-anonymity in the spatial domain (i.e., $k_{space} = 2$ at x_2) without downgrading. Assume the policy requires 4-anonymity. At time t_2, our Analyzer determines that 4-anonymity cannot be achieved without downgrading, and therefore performs temporal or spatial cloaking. The cloaking is illustrated as an expansion of the region (in dashed boxes) up to the point where at least 4 distinguishable GPS data reside within the region. As a result of downgrading, the x-coordinates $[x_2, x_3]$ with time interval $[t_2, t_4]$ or x-coordinates $[x_2, x_4]$ with time interval $[t_2, t_3]$ satisfy the 4-anonymity constraints.

However, in the face of a more capable attacker, other downgrading solutions must be applied. For example, there is a privacy risk present with an adversary that follows a specific object because consecutive GPS samples contain temporal and spatial correlation. Thus, paths of individual vehicles can be reconstructed by an attacker capable of using target tracking algorithms. Therefore, for the same Anti-Purpose (i.e., the capability to track individual vehicles), but in the context of more capable adversaries, other downgrading solutions must be applied by the Analyzer. One such solution is known as Maximum Time To Confusion (MTTC) [17]. MTTC is the maximum time for which an attacker is allowed to track an individual vehicle with a given tracking uncertainty. Tracking uncertainty measures how unlikely it is that a set of samples is associated with a particular vehicle. Under these circumstances, our downgrader needs to process a stream of GPS samples to maintain the tracking time bounds with a given allowable maximum time to confusion (MTTC) and an associated uncertainty threshold. Assume MTTC is set to $(t_5 - t_1)$ time units and that at time t_1 a potential attacker was sufficiently confused, i.e., for a given uncertainty threshold $Th_{uncertainty}$, an attacker could not determine the next location of the vehicle with tracking uncertainty lower than $Th_{uncertainty}$. The downgrading mechanisms must only reveal GPS samples when (i) the time since the last point of confusion (i.e., t_1) is less than MTTC or (ii) the current tracking uncertainty is above $Th_{uncertainty}$.

The Traffic Alert and Collision Avoidance System (TCAS) is a family of airborne devices that function independently of the ground-based air traffic control (ATC) to reduce the risk of midair collisions between aircraft. While it has proven its value as an augmentation of ground-based ATC over the past 20+ years, it has long been recognized that a more robust capability than TCAS can provide is needed to support the Free Flight concept that is envisioned for future airspace management. The limitations of TCAS result from the paucity of information that can be exchanged between aircraft over the limited communication bandwidth available, requiring it to try to extrapolate a 4D trajectory prediction using a zeroth-order dead reckoning (DR) algorithm. The quality of the trajectory projection could be significantly improved by also exchanging time-stamped motion state vector measurements (i.e., GPS data with velocity and acceleration vectors along with the aircraft attitude [roll, pitch, and yaw] and attitude rates-of-change) between aircraft, allowing use of a second-order DR algorithm. Such information (collectively referred to as "TSPI" — for Time-Space-Position Information) is available from the Inertial Navigation System (INS) that is an integral part of every commercial aircraft flying today. However, some military aircraft have a security concern about sharing that information with the general public. That concern is related to the potential for such information to be used (along with the measured strength of a radar "ping") to calculate an entry in the Radar Cross Section (RCS) database for that aircraft — i.e., a measure of how detectable that aircraft is. For many military aircraft, the values in the RCS database are classified information.

In this example, the challenge is to determine if there is a way to downgrade the TSPI from the INS so that the actual data transmitted will still allow the Purpose (trajectory prediction sufficiently accurate to avoid mid-air collisions between aircraft engaging in Free Flight), but will not enable its Anti-purpose (used to determine RCS values). We can formulate these Purpose and Anti-purpose in our framework using TSPI parameters like timestamp, position, velocity, acceleration, attitude and attitude rates, all concepts specified accordingly in KB models. The specific requirements on TSPI parameters differ for Purpose and Anti-purpose (e.g., Purpose can be achieved, albeit suboptimally, with attitude and attitude rate parameters redacted from the TSPI message, though they are needed for the Anti-purpose). This difference can be used to define possible downgrading solutions.

In summary, it is not our goal to introduce new downgrading mechanisms or algorithms. The goal of our work is to provide a framework in which existing downgrading solutions can be formalized and made amenable to a formal analysis process to determine whether, for a given problem with defined Purpose and Anti-purpose, existing downgrading algorithms can be used (possibly together) to achieve the goals.

VIII. Conclusion

A policy-based approach to enforcing privacy policies while enabling desired activities has the advantage of being

agile and flexible. Policies are described at a high level of abstraction in terms of enabled and forbidden activities, and the proposed Analyzer uses reasoning techniques to ensure the right balance between privacy and data sharing. A well-founded, automated solution to PBD is relevant to a wide range of applications that currently either lack a principled approach to privacy, or if privacy is addressed at all, typically provide only very specific solutions that cannot be reused.

The existing ONISTT Analyzer, on which our approach is based, is general enough to handle simple versions of the downgrading problem. We are currently extending our semantic models to include various downgrading operations and we are improving our Analyzer algorithms to handle the more complex constraint solving encountered in many downgrading scenarios. The technical challenges that we must address in the future to achieve a general downgrading tool are described below.

Downgrading solutions will often involve compositions of downgrader operations. Thus, we must develop an algebra for downgrading operations that the Analyzer can use to compose new downgraders as the need arises. The reasoning techniques needed to solve downgrading problems will also include a range of mathematical constructs, from simple matching to more complex constraint solving. Another complication stems from the fact that we do not limit our approach to static data sets, but intend to provide downgrading solutions for streaming data and for situations where downgrading needs to change dynamically, as receivers, data or policy change, or new insights into background knowledge requires more stringent downgrading. Data and error distribution management poses another technical challenge. We aim to provide a formal basis to capture different error distributions and reason about compositions of distributions. Finally, the task of automated generation of downgrader specifications is new, and possible in our framework due to its rich semantic models. The Analyzer and the Downgrader Specification Generator together provide a full circle set of tools to solve practical downgrading problems.

Acknowledgements. This work was partially sponsored by the Office of the Deputy Undersecretary of Defense for Readiness and by NAVSEA Indian Head under contract N00174-08-C-0057[3].

REFERENCES

[1] B. Fung, K. Wang, R. Chen, P. Yu, "Privacy-preserving data publishing: A survey on recent developments," *ACM Computing Surveys*, 2010.

[2] N. Adam and J. Worthmann, "Security-control methods for statistical databases: a comparative study," *ACM Computing Surveys*, vol. 21, no. 4, pp. 515–556, 1989.

[3]Some of the observations expressed in this paper are derived from those efforts, but are solely the views of the SRI International authors.

[3] B. Chen, D. Kifer, K. LeFevre, and A. Machanavajjhala, "Privacy-preserving data publishing," *Foundations and Trends in Databases*, vol. 2, no. 1-2, pp. 1–167, 2009.

[4] Z. Yang, S. Zhong, and R. Wright, "Anonymity-preserving data collection," in *Proc. 11th ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2005, pp. 334–343.

[5] C. Dwork, "Differential privacy," *Lecture Notes in Computer Science*, vol. 4052, pp. 1–12, 2006.

[6] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[7] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 3, 2007.

[8] D. Hanz, G. Denker, and A. Gehani, "Purpose-aware assured information sharing," *Submitted for review*, 2010.

[9] L. Bunch, J. M. Bradshaw, and C. Young, "Policy-governed information exchange in a U.S. Army operational scenario," in *Proc. Demonstration Track of the 2008 IEEE Conference on Policy*, 2008.

[10] R. McGraw, "Risk-Adaptable Access Control (RAdAC)," in *Presented at Privilege Access Management Workshop, September 1-3, NIST*, 2009, http://csrc.nist.gov/news_events/privilege-management-workshop/radac-Pa%per0001.pdf.

[11] S. Kumar and K. Moore, "The evolution of global positioning system (GPS) technology," *Journal of Science Education and Technology*, vol. 11, no. 1, 2002.

[12] R. Ford, D. Hanz, D. Elenius, and M. Johnson, "Purpose-aware interoperability: The ONISTT ontologies and analyzer," in *Simulation Interoperability Workshop*, 2007.

[13] D. Elenius, R. Ford, G. Denker, D. Martin, and M. Johnson, "Purpose-aware reasoning about interoperability of heterogeneous training systems," in *Proc. International Semantic Web Conference*, 2007.

[14] D. Elenius, D. Martin, R. Ford, and G. Denker, "Reasoning about resources and hierarchical tasks using OWL and SWRL," in *Proc. 8th International Semantic Web Conference (ISWC2009)*, October 2009.

[15] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "OWL web ontology language reference," October 2004, http://www.w3.org/TR/owl-ref/.

[16] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A semantic web rule language combining OWL and RuleML," May 2004, http://www.w3.org/Submission/SWRL.

[17] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson, "Virtual trip lines for distributed privacy-preserving traffic monitoring," in *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, 2008, pp. 15–28.