Chapter 11

A SURVEY OF RFID DATA PROCESSING

Charu C. Aggarwal
IBM T. J. Watson Research Center
Hawthorne, NY 10532, USA
charu@us.ibm.com

Jiawei Han
University of Illinois at Urbana-Champaign
Urbana, IL, USA
hanj@cs.uiuc.edu

Abstract

Radio Frequency Identification (RFID) is a new technology which allows a sensor (reader) to read, from a distance, and without line of sight, a unique product identification code (EPC) associated with a tag. Such tags are very useful in inventory management and logistics, because they can be used in order to track the movement and locations of large volumes of items in a cost effective way. This leads to massive streams of noisy data, which can be used in the context of a variety of data management and event processing algorithms. The use of RFID also has a number of privacy challenges associated with it, because a tag on an item being carried by a person, also becomes a unique location tag for that person. Therefore, methods need to to be designed to increase the privacy and security of RFID technology. This chapter will provide a broad overview and survey of a variety of RFID data management, mining and processing techniques. We will also discuss the privacy and security issues associated with the use of RFID technology.

Keywords: RFID Data, RFID Mining

C.C. Aggarwal (ed.), *Managing and Mining Sensor Data*, DOI 10.1007/978-1-4614-6309-2_11,

1. Introduction

RFID technology is a recent sensor technology, which allows *uniquely identifiable* tags to be read from a distance with the use of a sensor reader. RFID sensor technology is useful for tracking *very large volumes of items with specific identifiability* in a cost effective way. When combined with more sophisticated sensors transmitting real-time information and internet-enabled web services, this allows real-time connectivity and tracking of information about a wide variety of objects in daily life. The capability has been recognized in sensor computing as a paradigm shift in how objects are tracked in a ubiquitous manner, and is generally referred to as the *Internet of Things* [8].

At the most basic level, the definition of Radio Frequency Identification (RFID) is as follows: *RFID is a technology which allows a sensor (reader) to read, from a distance, and without line of sight, a unique product identification code (EPC) associated with a tag* [34]. Thus, the unique code from the tag is transmitted to one or more sensor reader(s), which in turn, transmit(s) the readings to one or more server(s). The data at the server is aggregated in order to track all the different product codes which are associated with the tags. *Passive RFID Tags* do not require an onboard battery, and can typically be read from a distance of a few centimeters to a few meters. Passive tags are typically powered by the radio signal that reads them. On the other hand, *active tags* come equipped with an onboard battery, which provides larger read ranges. If the tags are equipped with a sufficiently powerful antenna, it is also possible for them to transmit very long range signals, such as enabling the readability of the signal from satellites. While active tags have larger ranges, they come at a larger unit cost, and also have limited life spans. For most retail applications, passive tags are used in order to minimize the costs associated with the infrastructure. The primary fixed cost of such an infrastructure is embedded in the hardware and software associated with the sensor readers, whereas the variable costs of this system are associated with the RFID tags, each of which needs to be affixed to a tracked item. Typically, the number of sensor readers being used in large scale applications (such as retail tracking) is relatively small compared to the number of objects being tracked. For example, in a typical retail application, each tracking point will have a small number of sensor readers, which keep track of a very large volume of RFID tags passing through that point.

RFID sensors vary from conventional sensor technology in a variety of ways. With most sensors (whether mobile or stationary), the objects or readings which are sensed are not done so *actively*, and are usually not

powered by a sensor reader. Conventional sensors are used in conjunction with a battery for tracking (and wireless transmission of) readings such as ambient sound, temperature, light, videos, pressure, locations, or other objects, which are then transmitted by the sensor itself. In the case of RFID, the key is to identify specific information in the tag, by powering it with a sensor reader. This specific information in the tag is also known as *Electronic Product Identification Code (EPC)*. The uniqueness of identification code, and the cost-effectiveness of the tag, allows the simultaneous tracking of a large number of objects, since the presence of an object at a particular location can be associated with the identification code on its tag.

While the core idea of RFID technology is not new, and dates back to World War II for distinguishing between friendly and enemy aircrafts [53, 50], recent years have seen the emergence of a new *stripped down* version of the tag, which lacks a power source or antenna, and does little more than provide a unique identifier. Unlike regular sensors, such tags are *extremely inexpensive*, cost no more than a *few cents* each, and can easily be constructed for large scale applications. Some of the earliest discussions on the the rapid advancement of RFID technology to such large scale applications may be found in [56, 71]. The trend towards, *smaller*, *unobtrusive*, and *inexpensive* tags is exemplified by the following:

- Zebra has developed a print engine, which can embed an RFID transponder directly into a product label [16].

- Hitachi has developed an extremely tiny RFID tag, known as the μ-chip, which can be directly embedded into photocopier paper [81]. This can be used for document tracking.

These different kinds of developments suggest the continuing miniaturization of RFID technology across different domains. Furthermore, these developments also suggest that the applications of RFID technology go well beyond retail applications. It is important to note that the complexity of an RFID tag can be fairly flexible, depending upon the problem domain. If desired, it is possible to incorporate sensing into RFID technology [71, 72] with the use of onboard sensors that generate data dynamically. For example, an RFID tag may incorporate a temperature sensor (for perishable goods), or a passive force sensor, which can return information about the possible damage to a product, if it is dropped. Such tags are typically *active* RFID tags (with an onboard battery), and they are typically more expensive than *passive* tags, which are powered by a sensor reader, and return only the EPC. The par-

ticular choice of the tag depends upon the application domain, and an acceptable price-point for the tag in that application domain.

Thus, the broad flexibility in the functionality of RFID tags, makes them widely applicable to different problem domains. Examples of such domains are as follows:

- **Retail Applications:** In retail applications, RFID tags are associated with the products, and fixed sensor readers at particular locations are used in order to track the movement of products. The technique can be used for real-time inventory tracking. Alternatively, active shelves can be used in order to determine product availability.

- **The Internet of Things:** Ubiquitous computing, which is also referred to as the *internet of things*, has been identified as a key trend in recent years, in which information about objects is continuously tracked with the use of sensor technology. Along with a host of other advances in embedded sensor technology, RFID has been identified to be one of the key enabling technologies towards this trend [74]. The application of RFID technology for enabling such ubiquitous computing requires the coupling of basic RFID technology with the relevant web and internet services for allowing ubiquitous tracking. In particular, the ability to simultaneously identify a large number of objects uniquely in a cost-effective way with RFID tags has been a driving force in this direction.

- **Medical Applications:** RFID has increasingly found acceptance in a variety of pervasive healthcare applications [67]. For example, the tags may be associated with the patient medical history. This can be useful for automated tracking of patient medical history. For RFID-enabled healthcare asset management, major healthcare equipments, such as wheelchairs or other medical equipments are RFID-tagged, so that health-care experts can locate any asset in real-time. This can also help increase emergency room safety in addition to time saving [68].

- **Payment Systems:** RFID tags are used as credit-card like payment tokens that contain a serial number. When the tag is scanned for a payment, the reader transmits the number over a network to a remote computer, which is authorized to debit the money from the consumer's back account. An example of such a payment system is Texas Instruments's *Speedpass*, pay-at-pump system, which was introduced in Mobil stations in the mid-nineties.

- **Access Control:** The transmitters which are mounted on vehicles emit a signal which is read by the sensors at the tolls. This is used to keep track of the number of accesses through the toll, and also control the access of the vehicle through the toll. A popular such system is the New York's *EZ–Pass*, which is equipped with a 921.75 MHz semi-passive tag. Access control to commercial buildings and installations for employees and workers is often managed with the use of cards with RFID chips embedded in them.
- **Animal Movement:** Animals can be implanted with RFID tags in order to trace their movement. Alternatively, pet owner information can also be implanted on the RFID tag. Both kinds of tags can be useful in locating a lost pet, or in tracking the patterns of movement of wild life. More sophisticated tags (equipped with GPS receivers and transmitters) have been used in order to transmit signals that can be picked up by satellite, and have been used to track aquatic animals and other wild life.
- **Library Tracking:** RFID technology can be be used in order to automate the tracking of items which are checked out from the library by patrons. Such RFID tags also serve as security devices with the use of exit sensors, which track items that are being removed from the library, but have not been checked out. For example, several libraries such as the Santa Clara City library in California, the University of Nevada, Las Vegas library, and the Eugene, Oregon public library have already tagged every book, tape, CD, or other item in their collection [55].
- **Airline Luggage Management:** In this case, the RFID chips are attached to the luggage tags. Therefore, by placing sensor readers at strategic locations, it is possible to track the movement of luggage. This has been shown to be very useful in reducing lost luggage [54, 66].
- **Automobile Immobilizers:** Some of the newer car models have keys which contain an RFID tag. This key is authenticated by the steering column, and is required for vehicle operation. Such immobilizers typically have a small read range of a few centimeters, and operate in the low frequency end of the electromagnetic spectrum. Such systems have been widely credited with greatly reducing auto theft.

Even though RFID technology has been around for many years, its use for *large scale applications*, has only recently found widespread acceptance. The massive nature of RFID data is associated with numerous

challenges from the perspective of mining and analysis. These challenges are as follows:

- The volume of data associated with RFID can be extremely large, because of the large number of tags which may be tracked by a single reader. Furthermore, in some applications, the number of readers may also be quite large, which leads to a *high fan-in and hierarchical* organization of the underlying sensor network [23]. Such a system poses numerous challenges, because successive hierarchical aggregation of streams from different nodes of the network can lead to a overwhelming amount of data at the higher level nodes. It has been argued in [23] that a uniform stream-oriented query processing approach at all levels of the hierarchy works best in such systems.

- The data can be very *noisy* and *redundant*, with many tags being completely dropped, and others being read by multiple readers at multiple instants, resulting in tremendous redundancy of the representation. Furthermore, the large volume of the data makes the process of cleaning much more challenging. Therefore, effective methods need to be designed to compress and clean such data. The cleaning process can be rather expensive, and challenging, especially when near real-time responses to location queries are required.

- Many applications such as high level semantic event detection can be extremely challenging because of the high volume of the stream, and the real time nature of such applications. The noise and errors in the underlying data can lead to additional ambiguities during the event detection process.

- RFID deployments lead to a number of privacy concerns, because tags are uniquely identifiable by readers. Therefore, by carrying a tag attached to clothing, it may be possible to covertly track people without their knowledge. A variety of methods need to be designed in order to increase the privacy and security aspects of RFID technology.

- As with any sensor infrastructure, RFID technology and readers vulnerable to partial or complete system failures. Such failures can also lead to challenges in data processing, because data which is not collected will always be missing from the database. If the missing data is not explicitly accounted for by the underlying data analytics, it may lead to inaccurate inferences, because the missing

data may be interpreted as the absence of a particular item, rather than a system failure.

The chapter will discuss the processes involved in the storage, management and cleaning of RFID data. The chapter is organized as follows. In the next section, we will discuss the process of RFID Data Cleaning and compression. In section 3, we will discuss issues in the data management and warehousing of RFID data. An important goal of tracking RFID data is to use it for detecting interesting semantic events in the data, especially in real-time streaming scenarios. Section 4 discusses methods for event detection from RFID data streams. Section 5 discusses issues related to privacy and security of RFID data. Section 6 contains the conclusions and summary.

2. Raw RFID Data Cleaning and Compression

A variety of middleware architectures are used in order to collect and process RFID data [11, 22, 33, 37, 80, 68]. RFID data, by its very nature, is extremely noisy, incomplete and redundant because of cross-reads from multiple sensor readers. For example, it has been shown [38, 65] that a large fraction of the readings in RFID streams are essentially dropped. It has been estimated in [38, 65], that as many as 30% of the sensor readings are lost (i.e. the tag identifiers do not appear at all), because of the reader unreliability. Therefore, RFID middleware systems are used in order to correct for dropped readings. A commonly used method in many data cleaning systems [33, 80] is to use a temporal smoothing filter, in which a sliding window over the reader's data stream interpolates for lost readings from each tag within the time window. This approach provides each tag more opportunities to be read within the smoothing window. This reduces the number of distinct tags which are lost, because they will show up in one or more tag readings, when the window size is increased. Typically, the window size is fixed, as in [22], and the smoothing is performed on the basis of the readings which are received within this fixed window.

It has been observed in [36], that the choice of window size can be a critical parameter, which leads to different tradeoffs between false positives and false negatives. Using a window size which is too small will lead to missed readings (or *false negatives*), because the tag has fewer opportunities to be scanned by the reader. On the other hand, a larger window size will cause *false positives*, because it will lead to scanned readings, even after the tag has moved out of the reader's detection range. The work in [36] proposes *SMURF (Statistical sMoothing for Unreliable RFid data)*, which is an adaptive smoothing filter for raw RFID data streams.

This technique determines the most effective window size automatically, and continuously changes it over the course of the RFID stream, depending upon the underlying readings. One characteristic of this approach is that it does not expose the smoothing window parameter to the particular application at hand. This makes the approach much more flexible in different scenarios.

The approach proposed in [36] views RFID readings as *unequal probability random sample* of tags in the physical world. Therefore, the tradeoff between reader unreliability and tag dynamics can be explored in a principled and statistical manner. Furthermore, the approach can be used to clean both "single-tag" and "multiple-tag" readings. In the multiple tag case, it is assumed that single readings do not need to be tracked. For example, a store may only need to track when the number of items of a particular type falls below a given threshold. For the single tag case, binomial sampling methods are used for the cleaning process. For the multi-tag case, the aggregate signal over a tag population is cleaned with the use of Horvitz-Thompson estimators.

One characteristic of effective cleaning methods is to use *declarative methods* in the cleaning process [38, 39, 36]. The broad idea is to specify cleaning stages with the use of high-level declarative queries over relational data streams. Once this is done, the system can translate the queries into the required low level operations. Such an approach is useful in helping programmers avoid writing low level interaction code, by specifying the queries at the high level. Furthermore, such an approach makes the system data- and device-independent, and the code does not need to be changed if the underlying device fails, or is upgraded.

We note that the middleware approach to RFID data cleaning performs all the processing on the data upfront, before applying any of the data querying or analytical methods on it. However, different applications may define the anamolies or corrections on the same data set in a different way. Therefore, the method in [59] introduces a deferred approach for detecting and correcting RFID anamolies. Each application uses declarative sequence-based rules in order specify, detect, and correct relevant anamolies. We note that this approach is generally different from the methods proposed in [22, 36], which make the cleansing process application-independent. Clearly, both approaches have their own advantages in different scenarios. The generally accepted principle [22] is that the separation of the middleware from the applications is a desirable goal, because of the diversity of the applications in which such data could be used, and the network limitations of the underlying readers.

The actual process of cleaning may be much more complicated in an application containing thousands of readers and millions of tags. In such a case, the process of cleaning may incur tremendous *costs* associated with the entire process. These costs may be associated with either the cleaning plan itself, or in the misclassification associated with the cleaned data records. Therefore, a method for cost-conscious cleaning of massive RFID data sets has been proposed in [27].

The work in [27] assumes that three different kinds of inputs are available:

- A set of tag readings are available, which form a representative sample of the possible set of readings. Each reading is associated with a correct location of the tag, contextual information, area conditions, and tag protocol.
- A set of cleaning methods with associated per-tuple cleaning costs are specified.
- A per-tuple mis-classification cost is specified, which may be constant, a function of the tag reading and incorrectly assigned location.

The goal of the cost-sensitive approach is to learn a cleaning plan that identifies the conditions (feature values) under which a specific cleaning method or a sequence of cleaning methods should be applied in order to minimize the expected cleaning costs, including error costs. The work in [27] proposes a cleaning method which dynamically adjusts the probability of tag-presence based on the last observation. This is essentially a *Dynamic Bayesian Network (DBN)* approach. It has been shown in [27] that such an approach can outperform or complement methods which are based on smoothing windows. One advantage of DBN-based cleaning is that it does not require the use of recent tag readings (as in a window-based method), and it also gives more importance to recent readings, since the probability of tag-presence is continuously adjusted by the incoming tag readings.

A method called *StreamClean* has been proposed in [46], which uses *global integrity constraints* in order to clean the data. The core idea in *StreamClean* is that the tuples in a data stream system are not random, but are often related to one another, according to application-specific criteria. An example of such an integrity constraint provided in [46] can be as follows:
A car parked in the garage at time $t_e < t$ must either have exited in (t_e, t), or it must still be parked at time t.
In essence, the approach in *StreamClean* requires the specification of

user-stated properties that are true about the data. The system then uses these properties in order to insert missing tuples or correct conflicting tuples. In the event of groups of conflicting tuples, a probability of correctness is assigned to each tuple. Thus, the *StreamClean* approach transform the data to a probabilistic representation, in which explicit probability values are assigned to tuples. The approach then transforms constraints on the tuples into constraints on the underlying probability values. This also allows the possibility of *soft constraints*, in which a probability of a fact being correct is specified, rather than a *hard constraint*, in which the fact is deterministically known to be correct. The *StreamClean* method uses a non-linear optimization method, where the objective is to determine a probability assignment that maximizes entropy while satisfying the integrity constraints. The intuition behind maximizing entropy [32] is that in the absence of additional knowledge, the underlying solutions should be as uniform as possible. For example, the use of entropy maximization results in the explicit assumption, that in the absence of stated constraints, the probabilities of different input tuples are independent of each other. While this may not necessarily be true in all solutions, it is the most reasonable assumption to make in the absence of prior beliefs about such tuples.

It has been observed in [29] that RFID data exhibits a considerable amount of redundancy because of multiple scans of the same item, even when it is stationary at a given location. In practice, one needs to track only interesting movements and activities on the item. This is an issue which we will discuss in some detail in the next section on data management and warehousing. RFID tag readings also exhibit a considerable amount of *spatial redundancy* because of scans of the same object from the RFID readers placed in multiple zones. This is primarily because of the spatial overlap in the range of different sensor readers. This provides seemingly inconsistent readings because of the inconsistent (virtual) locations reported by the different sensors scanning the same object. It has been observed in [15] that the redundancy is both a blessing and a curse. While the redundancy causes inconsistent readings, it also provides useful information about the location of an object in cases, where the intended reader fails to perform its intended function. In addition, it has been observed in [15], that a considerable amount of background information is often available, which can be used in order to enhance accuracy. This background information is as follows:

- Prior knowledge about tagged objects and readers can be used in order to improve accuracy.

- Information about the constraints in the underlying application, such as the maximum capacity of a room or shelf, can be used in order to improve accuracy.

The work in [15] proposes a Bayesian inference framework, which takes full advantage of the duplicate readings, and the additional background information in order to maximize the accuracy of RFID data collection.

A different method, proposed in [52], is to use a *KerneL dEnsity-bAsed Probability cleaning method (KLEAP)* to remove cross-reads within a sliding window. The method estimates the density of each tag using a kernel-based function. The idea is that the most relevant reader will have a much larger number of objects in a similar position as this object. Therefore, the reader corresponding to the micro-cluster with the largest density will be regarded as the relevant position of the tagged object in the current window. The reads which are derived from the other readers will be treated as cross-reads.

3. RFID Data Management and Warehousing

Some of the earliest work on temporal management of RFID data was proposed in [68]. This work develops a *Dynamic Relationship ER (DRER) Model* for temporal management of RFID data. This system is built on top of the ER model with relatively few extensions. The technique maintains the history of events and state changes, so that complex queries can be supported. A rules-based framework is used to transform business logic data into user configured rules. In addition to location, another concept which is introduced is that of *containment*. Containment implies a hierarchical relationship between a set of objects. For example, a pallet may be loaded with cases, and both the pallet and the cases would have their own separate EPCs.

The RFID data contains two basic categories of data, corresponding to *static* and *dynamic* data. The static data is related to commercial entities such as location information, product level information, and serial information. There are two kinds of dynamic data: (a) The first corresponds to *instance data* such as serial number and the date of manufacture; and (b) The second corresponds to *temporal data* such as location observations and temporal changes in the containment of objects.

The second kind of temporal data are captured through EPC tag readings, and is related to the movement of products. The four primary kinds of entities which interact with one another in such a system are *EPC-tagged objects*, *readers*, *locations*, and *transactions*. These entities interact with one another, as object locations change, and entity containment relationships change as well. We note that even sensor (reader)

locations may change over time, as they are moved from one place to the other. Besides state changes, events are also generated in the interactions, including *observations*, when EPC tags interact with readers, and *transacted items*, when an object participates in a transaction.

The dynamic entity-relation model (DRER) is an extension of the ER model. In the ER model, all entities and relationships are static or current. In the RFID system, entities are static, but the relationships between them are dynamic. Thus, the only addition to the traditional ER model is the addition of a new kind of relationship, known as the *dynamic relationship*. There are two kinds of dynamic relationships, one of which generates events, and the other generates state history. An event-based dynamic relationship is associated with a single attribute known as `timestamp`, which represents the time at which the event occurred. On the other hand, a state-based dynamic relationship is associated with two attributes `tstart` and `tend` corresponding to when the state started and ended.

Thus, in the DRER model, we have three different static entities corresponding to *sensor reader*, *object*, and *location*. In addition, an entity called *transaction* may be used in order to represent business transactions, though we omit it in the discussion here for simplicity. Each of the static entities is associated with its own set of static entity tables. State-based dynamic relationships correspond to *sensor location*, *object location*, and *containment*. We note that each of these relationships are dynamic, and naturally have a starting and ending time. Event-based dynamic relationships occur at a particular instant, and may correspond to an *observation*, which is generated by a sensor reading an EPC tag. The different static and dynamic tables in the DRER model, together with their attributes are summarized below:

Entity	Type	Table Attributes
Sensor	Static	`SENSOR(sensor_epc, name, description)`
Object	Static	`OBJECT(object_epc, name, description)`
Location	Static	`LOCATION(location_id, name, owner)`
Observation	Dyn.	`OBSERVATION(sensor_epc, value, timestamp)`
Containment	Dyn.	`CONTAINMENT(epc, parent_epc, tstart, tend)`
Obj. Location	Dyn.	`OBJECTLOCATION(epc, location_id, tstart, tend)`
Sens. Location	Dyn.	`SENSORLOCATION(sensor_epc, location_id, position, tstart, tend)`

These tables can be used in conjunction with a variety of SQL queries in order to resolve interesting aspects about the RFID objects. Some examples [68] of such queries are as follows:

RFID Object Tracking Queries: The `OBJECTLOCATION` table carries

most of the information needed for formulating RFID location tracking queries. Since the starting and ending time for each object is included in the table, a query can be performed on an EPC in order to determine the history of locations for that objects. The precise time taken for an object to move from one location to another can also be derived because of the presence of the `tstart` and `tend` variables. This history of locations can be sorted temporally in order to provide the history of locations for any object. Similarly, missing objects at a location can be obtained from the `OBJECTLOCATION` table by comparing the objects at that location, with the set of objects at any time and location, where they were previously known to be complete. The precise formulation of these queries is provided in [68].

RFID Data Monitoring Queries: It is also formulate containment or observation queries for any particular snapshot by making use of the `CONTAINMENT`, `OBJECTLOCATION` and `OBSERVATION` queries. In addition temporal joins can be performed between different objects by formulating queries which examine the overlap between their `tstart` and `tend` variables. For example, recursive containment can be easily queried with this approach with the use of `CONTAINMENT` table, and temporal aggregation can be performed on the number of items which passed through a location at a given time, by making use of the `tstart` and `tend` attributes of the `OBJECTLOCATION` table. Thus, the tables supported by the DRER model are *expressive* and can support a wide range of SQL queries [68].

We note that in order to transform the noisy RFID into the high level semantic tables discussed above, which are consistent and non-redundant, a number of rules need to be defined. These rules correspond to *data filtering*, *location transformation*, and *data aggregation*. We note that many relationship tables (such as *containment* tables) are not *explicitly* specified in the RFID data, and they need to be inferred and aggregated, based on the observation patterns. A rule-based framework is proposed in [68] in order to automate the transformation of primitive events into semantically cleaner representations. For example, a *data filtering* rule can be defined to scan the data within a sliding window in order to determine if there are duplicates of the same event in multiple readers. One of these can then be dropped. Similarly, when a new location for an object is defined by a particular reader, the ending time stamp for the last location is updated to the current time. An entry is created in the `OBJECTLOCATION` table which a new starting time stamp, which is the current time. The ending time-stamp for the new entry is set to `UC (Until Changed)`. This is an example of a *location transformation rule*. An example of a *data aggregation* rule is one in which

when a set of pallets are loaded onto a track, the set of EPC readings for all the objects are inserted as the children of the EPC of the truck, in the `CONTAINMENT` table. Thus, an event detector continuously monitors the observation streams, and triggers actions which generate the corresponding data.

One challenge with managing RFID data, which was noticed in [68] was that RFID data typically have very large volume, which can lead to accumulation of large volumes of data. This can lead to slower queries and updates. An important observation about RFID data is that they typically have a limited life span, starting from the time it is first tagged, to the time when it is sold to the customer. Therefore, the database management approach in [68] partitions the data into an *active* set of RFID data, which corresponds to items which are frequently updated; and an inactive set of data, which corresponds to items that are no longer updated frequently. Since the majority of the data becomes inactive over time, this leads to much faster queries of the active data during its lifecycle.

3.1 Efficient Warehousing of RFID Data

A related, but somewhat different kind of RFID data management and warehousing has been discussed in [29]. This approach is designed towards finding the relevant paths of items in the RFID scenario. This process is also designed towards modeling the dynamic relationships such as *containment*, except that it does so not just for explicit containment, but also for items which move together. Also, the mapping relationships are modeled somewhat differently. The approach is also designed for tracking specific *measures* associated with the RFID items, which is typical in a data warehouse.

As in the case of [68], methods need to be designed to handle the massive redundancy of different types. These could be because of multiple readings of the same item from the same reader at multiple times. Consider the situation, where a typical reading from an RFID tag is of the form $(EPC, Location, Time)$. We note that the same tag may be read many times at the same location, even though no significant event may have occurred involving the time. As in [68], the only two readings which are significant are the first and last moment at which the items were read. The work in [29] uses two main kinds of compression.

- **Temporal Compression:** Multiple scans of the same code at the same (virtual) location can be compressed significantly. For example, if an item is loaded on an ship from one port to another, then the virtual location of the item corresponds to "*ship*" and all scans

of the item aboard the ship can be reduced to two scans. In practice, the location is associated with the identifier of a fixed sensor reader with a virtual location, such as "*ship*". These two scans correspond to the time that the item was loaded on the ship, and the time at which the item was removed from the ship. Therefore, by storing the times when the item was first moved to the vicinity of the reader and the time that it was moved away from the reader, all the relevant information is represented [29]. This allows us to represent the item in the form $(EPC, Location, TimeIn, TimeOut)$. We note that the $TimeIn$ and $TimeOut$ variables are similar to the `tstart` and `tend` variables proposed in [68] for maintaining the object location tables.

- **Group-based Compression:** In most real scenarios, RFID items often move together in groups or consignments. For example, all items which are loaded onto the ship stay together throughout the trip. Therefore, all the individual RFID of the items can be replaced by a single *generalized identifier*, or GID. In practice, groups of items may split or merge, as items are loaded at ports from different sources, or split into different destinations. Correspondingly, the generalized identifiers can be arranged hierarchically, in order to effectively represent these merges and splits.

One challenge with the use of RFID data with traditional data warehousing techniques, is that traditional warehousing methods do not properly consider the spatial links between different data records, which are essential in the RFID scenario. Therefore, traditional dare warehousing techniques may fail, when they are directly applied to RFID data. For example, consider the situation, where the cleaned RFID representation is of the form $(EPC, Location, TimeIn, TimeOut : Measure)$, where "Measure" could correspond to a value such as the quantity of the item present at the given location. Such a representation could be used in order to respond to queries such as the number of items which are present at a given location at any given period. However, it cannot be used to determine the number of items which moved from one location to another in a given period, at least with traditional data warehousing operations.

Therefore, RFID warehouses can be represented in the form of three different tables [29]: (a) an *info* table which contains location independent information about the items, such as its SKU, Product type etc., (b) a *stay* table which essentially contains all the set of facts in the form $(EPC, Location, TimeIn, TimeOut : Measure)$ (or in aggregate form as GIDs instead of EPCs), and (c) a *map* table which contains the links

between the different records of the fact table. The map table links together the different records in the table which form a path.

We note that the map table is the only additional information which needs to be maintained in the case of an RFID data warehouse. The RFID warehouse can be viewed as a multi-level database, in which the lowest level of representation are the raw RFID records, whereas the higher levels contain the cleansed and compressed records. In addition to the use of group-wise movements for compressing the data, a variety of other abstractions can be used for further compression. For example, if a minimum time granularity of one hour is required, then the set of movements and stays occurring in a single hour can be consolidated into a single movement. Similarly, the location can be specified at a higher level of granularity, and the sizes and the types of products can also be consolidated. This is because users are often interested in queries at much higher abstraction levels. Many path segments which are less important can also be eliminated and consolidated into a single movement. For example, for a store manager, the movement of items between shelves may not be important, and can either be eliminated or consolidated with some other movement. All of these operations significantly reduce the size of the representation, and make higher level query processing operations much more efficient.

Some other characteristics of the different kinds of tables such as the *info table*, *stay table*, and *map* table are as follows:

- The *info* table contains path-independent dimensions. Each dimension can have an associated concept hierarchy on which OLAP operations can be performed. For example, one could drill down on a particular product category and support aggregate queries on this category.

- The *stay* table contains the *TimeIn* and *TimeOut* information for the different products. In order to save space, this information is stored in terms of aggregated GIDs of items which move together, rather than the individual EPC values.

- The map table contains the hierarchy of GIDs in the data. Each entry is of the form $(gid, (gid_1 \ldots gid_r))$. This implies that gid points to $gid_1 \ldots gid_r$. We note that at the lower levels, gid_k could correspond to an individual EPC. The higher levels of the gid, are also labeled with locations, with one identifier corresponding to each location for items in the gid.

We note that the use of the gids, as maintained by the mapping table can provide a very efficient way to perform the queries, since each

individual gid may contain a very large number of items which have traveled together. In order to materialize the measures such as counts the algorithm does not need to access the counts of the individual EPCs.

It has been observed in [30] that the movement trails of RFID data form gigantic commodity flowgraphs, which represent the locations and durations of the path stages traversed by each item. The work in [30] proposes a method to construct a warehouse of commodity flows, which is also referred to as a *FlowCube*. Similar to OLAP, the model comprises cuboids, which aggregate the item flows at a given abstraction level. In this case, the measure of each cell is a commodity flowgraph, which captures the major movements and trends, as well as significant deviations from the trend in each cell. The flowgraph can also be viewed at multiple levels by changing the abstraction levels of path stages. The latter is achieved by performing simultaneous aggregation of paths to all interesting abstraction levels, In addition, path segments with low frequency and rarely occurring cells are removed from the representation.

It has been observed in [28] that a clustered path database, which is natural to RFID applications, can be naturally modeled as a *compressed probabilistic workflow*. Each location corresponds to an activity, and locations are linked according to their order of occurrence. A link between two activities has a probability, which represents the percentage of time that one of the activities occurred immediately after the other. A probabilistic representation of the workflow can also be used in the context of the FlowCube. The details of such a concrete probabilistic workflow are provided in [28].

4. Semantic Event Extraction from RFID Data Streams

The discussion so far has focussed on low level cleaning, event extraction and data management of RFID. However, in many applications, the events to be discovered are *high level semantic events*, as opposed to the primitive event of an object moving from one location to another. Such events are also referred to as *complex* events. The problem of event mining in RFID processing is related to previous research on complex event detection in active databases and high fan in sensor systems [1, 7, 13, 14, 18, 26, 62, 76, 70, 79]. In particular, the work in [62] discusses a high fan-in architecture for a sensor network, and shows how it can be used in order to process complex events by combining RFID data with other kinds of sensor readings and stored data.

An example of such a high-level semantic event discussed in [78] is the shoplifting example, in which the event corresponds to an item be-

ing picked up at a shelf and then being taken out of the store without being checked out. Clearly, a sequence of occurrence or non-occurrence of primitive RFID events can be used to determine the occurrence of a higher level semantic event. The problem of complex event extraction is probably one of the most critical ones in RFID event processing, because the purpose of tracking RFID data is essentially to determine different kinds of semantic events based on application-specific criteria. Therefore, an expressive and user-friendly language is required to support this class of queries for event processing. A language called *SASE* was proposed in [78] for complex event processing over RFID data streams.

The *SASE* event language is a declarative language that combines *filtering*, *correlation* and *transformation* of simpler events, in order to determine how they are correlated on the basis of time- and value constraints. The *SASE* language uses the following form in order to determine events:

```
EVENT <event pattern>
[WHERE <qualification>]
[WITHIN <window>]
```

For example, the shoplifting event pattern can be captured using the following construct [78]:

```
EVENT SEQ(SHELF-READING x, !  (COUNTER-READING y),
                  EXIT-READING z)
WHERE x.id = y.id ∩ x.id = z.id
WITHIN 12 hours
```

We note that the `EVENT` clause of the above contains a `SEQ` construct, which specifies a set of (primitive) events in a particular order. In this case, the construct detects a `SHELF` reading, followed by *the absence* of a `COUNTER` reading, and then followed by an `EXIT` reading. The `SEQ` construct turns out to be quite useful in the context of a wide variety of RFID queries, because of its ability to detect sequential combinations of basic events. Such sequential combinations form the core of event detection in complex RFID scenarios.

The basic constructs such as `SEQ` and negation are already available in the existing languages. However, in the context of RFID data, a number of new features are added by the work in [78], such as the use of parameterized predicates for correlating events via value-bed constraints. Sliding windows are used for imposing temporal constraints. Methods are also proposed for resolving the semantic subtlety of negation, when used together with sliding windows.

A query plan in the *SASE* language uses a subset of the following six operators in order to resolve the queries:

- **Sequence Scan and Construction (SSC):** While sequence scan and construction are technically different operators, they are always used together. The corresponding component is referred to as *SSC*. When a query contains the `SEQ` construct in the language, all the *positive* components of the `SEQ` specification are handled by the sequence scan and construction operators. Thus, a subsequence of the original specification is handled by this pair of constructs. Thus, the function of the *SSC* is to transform a stream of events into a stream of event sequences, each of which represents a unique match of specified *SSC* sub-sequence type.

- **Selection:** This is the commonly used operator in relational query processing. In this case, this operator is used to filter each event sequence by applying different predicates.

- **Window:** The window operator imposes the constraint of the `WITHIN` clause. For each event sequence, it checks if the temporal difference between the first and last events is less than the specified window T.

- **Negation:** The negation operator handles the negative components of a `SEQ` construct which have been ignored by SSC.

- **Transformation:** This operator converts each event sequence to a composite event by concatenating attributes of all the events in the sequence.

Another recent method for event processing with RFID data has been proposed in [9]. This method has the ability to query different readers for data in order to make key real time inferences for events. In addition, methods have been designed to work with the code embedded in the RFID tags for event processing. The EPC tags represents a string, in which different portions of the string correspond to different parts of the information about the product. Therefore, any algorithm needs to be able to work effectively in terms of deciphering the importance of different portions of the string for event processing. The approach in [9] shows how to extend an SQL-based query language in order to make it suitable for event processing in the context of RFID data. This can be an advantage in many scenarios, because users are often more familiar with SQL-like languages. Because of this, recent systems for event processing [19] have generally tried to work with extensions of the SQL language for event processing.

It has been observed [6] that stream event detection algorithms can be generally formulated as *pattern matching algorithms* over data streams.

Many of the traditional database operators for stream processing cannot effectively handle the detection of arbitrary event patterns. These techniques are quite effective for regular expression matching, though not quite as effective for pattern matching. The latter is much more important in the stream scenario. Therefore, the work in [6] proposes a method for matching arbitrary patterns in data streams in order to perform event detection. This work proposes a formal query evaluation model, NFA^b that combines a finite automaton with a match buffer. This is used to create query evaluation plans that can be executed over event streams. One characteristic of this approach is that it uses storage sharing of all possible pattern matches as well as in automaton execution to produce these matches. This results in more efficient query execution.

4.1 Probabilistic Event Extraction

It has been observed in [45, 47] that there is an inherent ambiguity in the cleaning and determination of high level events of RFID data. Since, the collection of RFID data is prone to errors, it is natural that such data is best represented by probabilistic databases as discussed in [17, 31, 77]. The importance of using probabilistic representations for event extraction in pervasive computing applications has been discussed in depth in [25], though the approach discusses the design of an *inference engine* for event extraction. In the context of RFID data management applications, it is more critical to design a *query processing engine* for probabilistic event extraction. Therefore, the work in [45] proposes a probabilistic event language *PeexL* for defining probabilistic events. An implementation of the approach is proposed in a system called Probabilistic Event EXtractor *(PEEX)*, a middleware layer on top of a relational database management system (RDBMS). The idea is that uncertainty propagates as events are aggregated into higher level events. For example, a `MEETING` event can be inferred from a sequence of `ENTERED-ROOM` events by different participants. However, if there is limited confidence in the `ENTERED-ROOM` events, then the confidence in the `MEETING` events will also be lower. The work in [45, 47] uses confidence tables in order to track the confidence of the different events and then aggregate these probabilities into higher level event probabilities with the use of the *PeexL* language. Another interesting probabilistic event processing system known as *Lahar* has been proposed in [61]. This approach uses a framework which is similar to the *Cayuga* system [18] for event processing, except that it is focussed on querying probabilistic representations of the underlying data.

5. Privacy and Security Issues with RFID Data

One challenge with the use of RFID technology is that the tags on the items can be tracked by sensor readers without the knowledge or consent of people carrying them. For example, the items bought in a store can be used in order to track people, as they move about in the world. This is particularly true for items such as shoes or clothing. This has lead to increasing privacy concerns about the large-scale use of such technology [24, 43, 51, 57]. For example, the *Consumers Against Supermarket Privacy Invasion and Numbering (CASPIAN)* protested against apparel manufacturer Benetton for planning to attach RFID tags to their products. This lead to a boycott of those products in 2003 [57, 82]. *CASPIAN* similarly criticized Tesco for conducting experimental trials of tags on a variety of its products [83] in 2005.

An additional troubling aspect of the tags is that they contain no information about their read-history. While tags can be scanned by anyone without the consumer's knowledge, there is also no way for the consumer to know that they have been scanned. The EPC contains a serial number, which is unique to a particular *instance of the* product item. Therefore, once a customer buys the product and carries it on their person, the product EPC becomes a unique identifier for the customer, which can be distinguished from a similar product bought by another customer. This information can be misused in a variety of ways:

- Individuals carrying tagged products can be tracked with the use of covert readers placed at different locations.
- Since the EPC also contains manufacturer information, it can be used in order to order to obtain competitive information about customer preferences without their knowledge.
- When tagged items move from one individual to the other, the transactions between different individuals can be tracked.
- Associations are often built up between tagged items and individuals in corporate information systems, as individuals move around with tagged items over time. When these items are discarded, such associations are typically not broken. If these items are then used for malicious or illegal purposes, then this can expose the individual to different kinds of liabilities with law enforcement.

In addition to the personal privacy threats, a number of threats are possible with the use of RFID data at the corporate level. A particular area of concern is the tracking of RFID data for the purposes of *corporate espionage*. Tagged objects in the supply chain make it easy

for competitors to routinely gather information about the activities of a business.

The use of RFID technology also has security consequences which go beyond simple privacy concerns. These are as follows:

- RFID technology is highly dependent on the use of radio signals which are easily jammed. This can open the system to a variety of infrastructure threats.
- It has recently been demonstrated [10], that RFID tags can be cloned to emit the same identification code as another tag. This opens the system to fraud, when the RFID tag is used for the purpose of sensitive tasks such as payment. This can also be used in order to make the function of an automobile immobilizer vulnerable to attack.

We note that privacy issues for RFID data can arise both during *data collection* and during *data management*, once the RFID data has been captured. For the case of data collection, the information is typically stolen through eavesdropping on either the tag or the reader signal. In this case, since the privacy concerns arise from the design of the tag itself, many of the issues need to be addressed by enhancement and modification of the underlying tag, with either hardware or software solutions, or a combination of both. On the other hand, in the case of data management, the privacy issues relate to the access control of the underlying data. We will discuss some of the different methods for privacy preservation both during data collection and management in the following subsections.

5.1 The Kill Command

The Auto-Id Center designed the "kill" command, which are intended to be executed at the point of sale. The kill command can be triggered by a signal, which explicitly disables the tag [63, 64]. If desired, a short 8-bit password can be included with the "kill" command. The tag is subsequently "dead" and no longer emits the EPC, which is needed to identify it. However, the killing of a tag, can sometimes be an impractical solution in cases, where the tags have a utility beyond the point of sale. Some examples are as follows:

- The tags are used for identification purposes in order to facilitate the repairs or returns for the underlying products.
- Many smart appliances use the tags for other purposes. An example discussed in [24] discusses the smart refrigerator which uses

RFID tags in order to identify expired food. Clearly, the killing of a tag at the point of sale would make this functionality useless.

Therefore, a number of methods have been proposed, which go beyond the "kill" command for the purposes of providing privacy protection. A slightly softer solution is to use a locking and unlocking mechanism for the tags [75]. For example, the tags could be locked at the time of check out in the store. The tags can then be unlocked with a meta-id provided by the consumer, along with an associated PIN. This approach has two primary disadvantages. One disadvantage is that the incorporation of smart technology makes the tag much more expensive. The second disadvantage is that it is impractical for consumers to manage meta-identifiers and PINs for all the different products that they may buy.

5.2 Cryptographic Solutions

A possible solution is to encrypt the code in a tag before transmission. However, such a solution may not be very effective, because this only protects the *content* of the tag, but not the ability to *uniquely identify* the tag. For example, the encoded tag is itself a kind of meta-tag, which can be used for the purposes of tracking. Another solution is to embed dynamic encryption ability within the tag. Such a solution, however, comes at a cost, because it requires the chip to have the ability to perform such an encryption computation. Another solution which has recently been proposed [40] is to perform the cryptographic computations at the reader end itself, and store the resulting information in the tags. This solution of course requires careful modification of the reader-tag protocols. A number of cryptographic protocols for privacy protection of library RFID activity are discussed in [55]. Some of the cryptographic schemes [44, 48, 58] work with *re-writable* memory in the tags in order to increase security. The tags are encrypted, and the reader is able to decrypt them when they send them to the server, in order to determine the unique meta-information in the tag. The reader also has the capability to re-encrypt the tag with a different key and write it to its memory, so that the (encrypted) tag signal for an eavesdropper is different at different times. Such a scheme provides additional protection because of repeated change in the encrypted representation of the tag, and prevents the eavesdropper from uniquely identifying the tag at different times.

5.3 Blocker Tags

An interesting solution for making it difficult to read tags in an unauthorized way is the use of *blocker tags* [41, 42]. Blocker tags exploit the collision properties of RFID transmission, which are inherent in this technology. The key idea is that when two RFID tags transmit distinct signals to a reader at the same time, a broadcast collision occurs, which prevents the reader from deciphering either response. Such collisions are in fact very likely to occur during the normal operation of the RFID infrastructure. In order to handle this issue, RFID readers typically use anti-collision protocols. The purpose of blocker tags is to emit signals (or spam) which can defeat these anti-collision protocols, thereby causing the reader to stall. The idea is that blocker tags should be implemented in a way, that it will only spam unauthorized readers, thereby allowing the authorized readers to behave normally.

Typically the anti-collision protocols which are used are also referred to as *singulation* protocols, which allow the tag reader to systematically explore all the tags in a certain order with the use of a *tree-walking protocol*, which singles out all the tags for scanning in a specific order. This is achieved by treating the binary code on each tag in the form of a binary tree, where each node in the tree is considered a prefix of the binary tree. The idea is that the reader has the capability to scan for tags containing only a particular prefix, and ask all other tags to remain "silent". Tags which contain that particular prefix, transmit their bit which comes just after that prefix. The algorithm starts at the root of the tree, and scans the first bit of the tags. In the event that both 0 and 1 is transmitted, then a collision will occur, which is detected. This means that both branches of the tree need to be explored, since there are tags which contain both a 0 and a 1 in the first. Clearly, a collision is quite likely to occur at the higher levels of the tree. On the other hand, if only a 0 is transmitted, then the left branch of the tree needs to be explored. Otherwise, the right branch of the tree is explored. This process is used to recursively traverse the portion of the tree which is relevant to the RFID tags being scanned. This recursive traversal finally reaches the leaves of the tree, at which point, the tags are recorded uniquely by the reader. It is clear that for a 96-bit Class 1 EPC tag, the portion of the tree which is explored by the reader is an extremely tiny fraction of the 2^{96} possible nodes in the tree, since the number of distinct tags being present would be much smaller than 2^{96}. In fact, the entire tree-size is too large to be explored by the tree-walking algorithm.

The blocker tag takes advantage of this property and forces (malicious) readers to explore the full tree of size 2^k, which would cause the

reader to stall. The idea is that for each query by the reader, the blocker tag *always* sends *both* and 0 and a 1. This means that the reader would have to recursively traverse the entire tree, an undesirable situation, which could cause the reader to stall. The blocker tag can be modified to block only those tags with certain prefixes, a process which is referred to as *selective blocking*. For example, the blocker tag may respond to the reader only for those prefixes which correspond to the left subtree of the root. This means that all RFIDs which start with a "0" are now protected or "blocked" from scanning. Thus, by carefully assigning prefixes to different products, it is possible to selectively block only certain kinds of products for the purposes of privacy protection. Alternatively, it may be possible to reset the prefix in a tag at check-out time, so as to move the tag from the unprotected zone to the protected zone, once it has been bought by the consumer. A blocker tag may either be carried by a consumer on their person (through active acquisition), or may be provided by a supermarket in a grocery bag, so as to prevent undesired scanning of the items bought by a consumer. Once the items are removed from the bag (for example, when food items are placed in a refrigerator), the tags can become usable again, since they have been removed from the vicinity of the blocker tag. The main drawback of blocker tags is that they only provide an "opt-out" mechanism, in which tags are active by default, and consumers must take the step of acquiring blockers in order to protect their privacy. Such opt-out mechanisms are very useful, when the tags only need to be blocked at certain times, places, or in the possession of certain people.

We further note that a *polite blocking* protocol can be implemented, which allows the readers to query the blocker tags, which tells them the portions of the tree that they should not traverse [41]. Thus, the blocker tag is being "polite" to the reader in telling it, which portions of the tree it is blocking. The tree-walking protocol can then be modified in order to not query those portions of the tree which are being blocked. Polite blocking is useful, when the environment may contain legitimate readers, which should not be made to inadvertently stall by the use of blocker tags. Since authorized readers are likely to follow the proper protocol, they will not be affected by the blocker tag. Furthermore, even if unauthorized readers use the proper protocol, they will be unable to access the tags of items with protected prefixes. This is the entire purpose of blocking.

The blocking approach can be considered a kind of passive jamming, and can be used both for privacy protection or for malicious purposes. When used for malicious purposes, it can be considered equivalent to a

"denial of service" attack, which prevents readers from performing their normal function by jamming them.

5.4 Other Privacy- and Security-Protection Methods

A number of privacy-protection mechanisms rely on the fact that the eavesdroppers are more likely to be at some distance from the tag. In this context, it was inferred by [75] that the greater threat to privacy arises from the eavesdropping of signals sent from the reader (which can be detected much further away), rather than reading the tag itself (which can be done only at a much closer distance). In fact, the IDs being read by the tree-walking protocol can be inferred merely by listening to the signals being broadcast by the reader. Therefore, it has been proposed in [75] to encrypt the signals being sent by the reader in order to prevent privacy attacks by eavesdropping of *reader* signals.

A recent approach proposed in [21] makes the observation that the legitimate readers are likely to be much closer to RFID tags, as compared to unauthorized readers which attempt to surreptitiously scan items. It is possible for a tag to detect the strength of the scanning signal, and change its behavior depending upon the distance. For closer readers, the full signal is transmitted, whereas for readers which are further away, only the information about the type of product is transmitted.

A variety of other methods are available to make RFID tags smarter for the purposes of privacy protection. For example, it is possible to modify RFID tags to cycle through a set of *pseudonyms* rather than emit a unique serial number [40]. Thus, the tag cycles through a set of k pseudonyms and emits them sequentially. This makes it more difficult for an attacker to identify the tags, because they may only be able to scan different pseudonyms of the tags at different times. Of course, if the attacker is aware of the method being used in order to mask the tag, they may try to scan the tag over a longer period of time, in order to learn all the pseudonyms associated with the tag. This process can be made more difficult for an attacker by increasing the time it takes for the tag to switch from one pseudonym to another.

Of course, the ability to modify the data in the RFID tags is also a security threat, when it is done by an adversary. Therefore, a natural solution is to password-protect the memory in the RFID tag. This is a challenge from an energy consumption perspective, since all cryptographic algorithms require a large amount of energy, and it would require an onboard battery (active tag) for enablement. In this context,

a number of methods, which have low energy requirements for these cryptographic solutions have been proposed recently [12, 20].

5.5 Privacy Issues in Data Management

In previous subsections, we addressed the privacy issues which arise as a result of eavesdropping on the tag or the reader. In this subsection, we will discuss the privacy issues which arise from the data management issues of the collected data. The general methods for privacy-preservation, such as k-anonymity, ℓ-diversity, t-closeness etc, are also applicable to the data which is captured using RFID technology [3]. The general goal of these methods is to reduce the fidelity of the captured data, so that aggregate inferences can still be derived from it, without compromising privacy.

A number of interesting challenges for privacy arise, when both people and objects are tagged, and the same people have access to the captured RFID data. Such a scenario arises in the context of an RFID Ecosystem constructed at the University of Washington [49, 74]. The most restrictive view to privacy would be one in which users only have access to their own data. While this assures complete privacy of a user, it also unnecessarily curtails the useful insights which one can obtain from such data. This is because events which occurred in the *proximity* of a given user at a given time should be accessible to the user, even if they do not directly relate to the user themselves. This is because such events could be observed by that user by virtue of their physical presence.

It has been observed in [49] that a natural access control policy to use in such a scenario is one in which the data to which a user can gain access is that which corresponds to events which occurred at times and places when and where the user was physically present. This policy is also referred to as *Physical Access Control (PAC)* in [49]. In a sense, such a policy provides a database view which augments people's memory of objects, places and people. It also naturally models the boundaries of people in everyday life. In addition, a user can also specify rules which can relax or restrict the access to data which concerns them. This provides a certain level of personal choice and flexibility in the privacy-preservation process.

The work in [60] further implements the broad principles of the PAC policy by designing a rule-based system, which can infer which information to release for a particular user. The system starts from PAC, and then uses a number of reasoning rules in order to make careful decisions about access control.

6. Conclusions and Summary

While RFID is a relatively old technology, its use for large scale applications has proliferated in recent years. This is because of technological advances in manufacturing, which have made the tags smaller and cheaper. The ability to manufacture a tag at less than 5 cents (per tag) has allowed their widespread use in a cost effective way. RFID data brings numerous challenges with it for the purposes of mining and analysis. RFID data is inherently noisy and redundant because of missed tag readings, or multiple readings of the same tag from different readers. Therefore, techniques need to be designed in order to make the process of reading more robust and reduce the redundancy in the underlying data. The massive volume of the RFID data also makes the process of warehousing and querying the RFID data much more challenging. Therefore methods need to be designed in order to represent the RFID warehouse in terms of the aggregated views of RFID items which typically move together. These aggregated views greatly improve the efficiency of data storage and querying. RFID data can be useful in detecting important semantic events from the underlying data streams. The existing work in active databases and sensor stream event detection can be further extended in a variety of ways to make it suitable to the RFID scenario. For example, methods have recently been designed for event processing in uncertain RFID data streams.

RFID data naturally leads to a number of privacy challenges, because of the association of people with tags, and the likelihood of monitoring people's location with such tags. The privacy issues with RFID data arise *both* during data collection and management. A number of methods such as the kill command, cryptographic protocols, and blocker tags have been designed for privacy protection during data collection. In addition, a number of methods for physical access control have been developed for preserving personal privacy during data management.

References

[1] R. Adaikkalavan, S. Chakravarthy. Snoopib: interval-based event specification and detection for active databases. *Data and Knowledge Engineering*, 59(1): pp. 139–165, 2006.

[2] C. C. Aggarwal. Data Streams: Models and Algorithms, *Springer*, 2007.

[3] C. C. Aggarwal, P. S. Yu. Privacy-Preserving Data Mining,: Models and Algorithms, *Springer*, 2008.

[4] C. C. Aggarwal, T. Abdelzaher. Social Sensing. *Managing and Mining Sensor Data*, Springer, 2013.

[5] C. C. Aggarwal, N. Ashish, A. Sheth. The Internet of Things: A Survey from the Data-Centric Perspective, *Managing and Mining Sensor Data*, Springer, 2013.

[6] J. Agrawal, Y. Diao, D. Gyllstrom, N. Immerman. Efficient Pattern Mining over Event Streams, *ACM SIGMOD Conference*, 2008.

[7] M. Akdere, U. Centintemel, N. Tatbul. Plan-based complex event detection across distributed sources, *VLDB Conference*, 2008.

[8] K. Ashton. That 'Internet of Things' Thing. In: *RFID Journal*, 22 July, 2009.

[9] Y. Bai, F. Wang, P. Liu, C. Zaniolo, S. Liu. RFID Data Stream Processing with a Data Stream Query Language, *ICDE Conference*, 2007.

[10] S. Bono, M. Green, A. Stubblefield, A. Juels, A. Rubin, M. Szydylo. Security Analysis of a Cryptographically Enabled RFID Device, *USENIX Security*, 2005.

[11] C. Bornhovd, T. Lin, S. Haller, J. Schaper. Integrating Automatic Data Acquisition with Business Processes Experiences with SAP's Auto-Id Infrastructure, *VLDB Conference*, 2004.

[12] B. Calmels, S. canard, M. Girault, H. Sibert. Low-cost cryptography for privacy in RFID systems, *Proceedings of IFIP CARIDS*, 2006.

[13] M. J. Carey, M. Livny, R. Jauhari. The HiPAC project: Combining active databases and timing constraints. In *SIGMOD Record*, 17(1), 1988.

[14] S. Chakravarthy, V. Krishnaprasad, E. Anwar, S. Kim. Composite events for active databases: Semantics, contexts and detection. *VLDB Conference*, pp. 606–617, 1994.

[15] H. Chen, W.-S. Ku, H. Wang, M.-T. Sun. Leveraging Spatio-Temporal Redundancy for RFID Data Cleansing. *ACM SIGMOD Conference*, 2010.

[16] J. Collins. Zebra Unveils RFID Label Maker. *RFID Journal*, September 25, 2003. http://www.rfidjournal.com/article/articleview/592/1/1.

[17] N. Dalvi, C. Re, D. Suciu. Query evaluation on probabilistic databases. *IEEE Data Engineering Bulletin*, 29(1), pp. 25–31, March 2006.

[18] A. Demers, J. Gehrke, M. Hong, M. Riedewald, W.M. White. Towards expressive publish/subscribe systems. *EDBT Conference*, 2006.

[19] N. Dindhar, B. Guc, P. Lau, A. Ozal, M. Soner, N. Tatbul. DejaVu: Declarative Pattern Matching over Live and Archived Streams of Events, *ACM SIGMOD Conference*, 2009.

[20] M. Feldhofer, S. Dominikus, J. Wolkerstorfer. Strong authentication for RFID systems using AES algorithm, *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, 2004.

[21] K. Fishkin, S. Roy. Enhancing RFID Privacy via Antenna Energy Analysis, *Tech. Memo IRS–TR–03–012*, Intel Research, Seattle, 2003.

[22] C. Floerkemeier, M. Lampe. RFID Middleware Design – Addressing Application Requirements and and RFID Constraints, *Joint Conference on Smart Objects and Ambient Intelligence*, 2005.

[23] M. J. Franklin, S. R. Jeffrey, S. Krishnamurthy, F. Reiss, E. Wu, O. Cooper, A. Edakkunni, W. Hong. Design Considerations of High Fan-In Systems, *CIDR Conference*, 2005.

[24] S. L. Garfinkel, A. Juels, R. Pappu. RFID Privacy: An Overview of Problems and Proposed Solutions, *IEEE Security and Privacy*, 3(3), 2005.

[25] M. Garofalakis, K. Brown, M. Franklin, J. Hellerstein, D. Wang, E. Michelakis, L. Tancau, E. Wu, S. Jeffery, R. Aipperspach. Probabilistic Data Management for Pervasive Computing: The Data Furnace Project. *IEEE Data Engineering Bulletin*, 29(1): pp 57–63, 2006.

[26] N. H. Gehani, H. V. Jagadish, O. Shmueli. Composite Event Specification in Active Databases: Model and Implementation, *VLDB Conference*, 1992.

[27] H. Gonzalez, J. Han, X. Shen. Cost-Conscious Cleaning of Massive RFID Data Sets. *ICDE Conference*, 2007.

[28] H. Gonzalez, J. Han, X. Li. Mining compressed commodity workflows from massive RFID data sets. *CIKM Conference*, 2006.

[29] H. Gonzalez, J. Han, X. Li, D. Klabjan. Warehousing and Analyzing Massive RFID Data Sets. *ICDE Conference*, 2006.

[30] H. Gonzalez, J. Han, X. Li. FlowCube: Constructing RFID FlowCubes for Multi-Dimensional Analysis of Commodity Flows. *VLDB Conference*, 2006.

[31] T. J. Green, V. Tannen. Models for incomplete and probabilistic information. *IEEE Data Engineering Bulletin*, 29(1), pp. 17–24, March 2006.

[32] S. Guiasu, A. Shenitzer. The principle of maximum entropy. *Mathematical Intelligence*, 7(1), 1985.

[33] A. Gupta, M. Srivasatava. Developing auto-id solutions using sun java system rfid software, October 2004.
http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/sjsrfid/RFID.html

[34] J. Han, J.-G. Lee, H. Gonzalez, X. Li. Mining Massive RFID, Trajectory, and Traffic Data Sets (Tutorial). *ACM KDD Conference*, 2008.
Video of Tutoral Lecture at: http://videolectures.net/kdd08_han_mmrfid/

[35] J. Han, H. Gonzalez, X. Li, D. Klabjan. Warehousing and Mining Massive RFID Data Sets. *ADMA*, 2006.

[36] S. R. Jeffrey, M. Garofalakis, M. J. Franklin. Adaptive Cleaning for RFID Data Streams, *VLDB Conference*, 2006.

[37] S. R. Jeffery, M. J. Franklin, M. Garofalakis. An Adaptive RFID Middleware for Supporting Metaphysical Data Independence. *VLDB Journal*, 17(2), pp. 265–289, 2008.

[38] S. R. Jeffrey, G. Alonso, M. Franklin, W. Hong, J. Widom. A pipelined framework for online cleaning of sensor data streams. *ICDE Conference*, 2006.

[39] S. R. Jeffrey, G. Alonso, M. Franklin, W. Hong, J. Widom. Declarative Support for RFID Data Cleaning, *Pervasive*, 2006.

[40] A. Juels. Minimalist Cryptography for RFID Tags. *Conference on Security in Communication Networks*, 2004.

[41] A. Juels, R. Rivest, M. Szydlo. The Blocker Tag: Selective Blocking of RFID tags for Consumer Privacy. *ACM Conference on Computer and Communication Security*, pp. 103–111, 2003.

[42] A. Juels, J. Brainard. Soft Blocking: Flexible Blocker Tags on the Cheap, *Workshop on Privacy in the Electronic Society (WPES 04)*, pp. 1–7, 2004.

[43] A. Juels. RFID Security and Privacy: A Research Survey, *IEEE Journal on Selected Areas in Communication*, vol. 24, pp. 381–394, Feb. 2006.

[44] A. Juels, R. Pappu. Squealing Euros: Privacy protection in RFIDenabled banknotes. *Proceedings of Financial Cryptography*, Springer–Verlag, 2003.

[45] N. Khoussainova, M. Balazinska, D. Suciu. Probabilistic Event Extraction from RFID Data, *ICDE Conference*, 2008.

[46] N. Khoussainova, M. Balazinska, D. Suciu. Towards Correcting Input Data Errors Probabilistically Using Integrity Constraints. *Fifth*

International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'06), 2006.

[47] N. Khoussainova, M. Balazinska, D. Suciu. Peex: Extracting probabilistic events from RFID data. *Technical Report 2007–11–02, Department of Computer Science and Engineering, University of Washington*, 2007.

[48] S. Kinoshita, F. Hoshino, T. Komuro, A. Fujimura, M. Ohkubo. Low-cost RFID privacy protection scheme. *IPS Journal*, 45(8), 2004.

[49] T. Kriplean, E. Welbourne, N. Khoussainova, V. Rastogi, M. Balazinska, G. Borriello, T. Kohno, D. Suciu. Physical Access Control for Captured RFID Data, *IEEE Pervasive Computing*, 6(4), 2007.

[50] J. Landt. The History of RFID. *IEEE Potentials*, October/November 2005.

[51] M. Langheinrich. A Survey of RFID Privacy Approaches. *Personal and Ubiquitous Computing*, Springer, 2008.

[52] G. Liao, J. Li, L. Chen, C. Wen. KLEAP: An Efficient Cleaning Method to Remove Cross-Reads in RFID Data Streams, *ACM CIKM Conference*, 2011.

[53] V. Krishnamurthy, S. Chawathe, S. Ramachandran, S. Sarma. Managing RFID Data, *VLDB Conference*, 2004.

[54] R. McManus. The End of Lost Luggage? RFID Slowly Coming to Airports, *ReadWriteWeb*, 2009.
http://www.readwriteweb.com/archives/the_end_of_lost_luggage_rfid.php

[55] D. Molnar, D. Wagner. Privacy and Security in Library RFID: Issues, Practices and Architectures, *CCS*, 2004.

[56] L. Ni, Y. Liu, Y. Lau, A. Patil. LANDMARC: Indoor Location Sensing using Active RFID, *Wireless Networks*, 10(6), pp. 701–710, 2004.

[57] M. Ohkubo, K. Suzuki, S. Kinoshita. RFID Privacy Issues and Technical Challenges, *Communications of the ACM*, 48(9), 2005.

[58] M. Ohkubo, K. Suzuki, S. Kinoshita. A cryptographic approach to "privacy-friendly" tags. *RFID Privacy Workshop*, 2003.

[59] J. Rao, S. Doraiswamy, H. Thakkar, L. S. Colby. A Deferred Cleansing Method for RFID Data Analytics, *VLDB Conference*, 2006.

[60] V. Rastogi, E. Welbourne, N. Khoussainova, T. Kriplean, M. Balazinska, G. Borriello, T. Kohno, D. Suciu. Expressing Privacy Policies using Authorization Views, *International Workshop on Privacy in UbiComp (UbiComp'07)*, 2007.

[61] C. Re, R. Letchner, M. Balazinska, D. Suciu. Event Queries on Correlated Probabilistic Streams, *ACM SIGMOD Conference*, 2008.

[62] S. Rizvi, S. R. Jeffrey, S. Krishnamurthy, M. J. Franklin, N. Burkhart, A. Edakkunni, L. Liang. Events on the edge, *ACM SIGMOD Conference*, 2005.

[63] S. E. Sarma, S. A. Weis, D.W. Engels. Radio-frequency identification systems. *CHES*, pp. 454–469, 2002.

[64] S. E. Sarma, S. A. Weis, D.W. Engels. RFID systems, security and privacy implications. *Technical Report MIT–AUTOID–WH–014*, AutoID Center, MIT, 2002.

[65] L. Sullivan. RFID Implementation Challenges Persist, All This Time Later. *Information Week*, October 2005.

[66] C. Swedberg. Reusable Electronic Baggage Tag Powered by RFID, *RFID Journal*, October 2010.

[67] W. J. Tu, W. Zhou, S. Piramuthu. Identifying RFID-embedded objects in Pervasive Healthcare Applications, *Decision Support Systems*, 46(2), 2009.

[68] F. Wang, P. Liu. Temporal management of RFID Data, *VLDB Conference*, 2005.

[69] F. Wang, S. Liu, P. Liu. Complex RFID Event Processing, *VLDB Journal*, 18(4), 2009.

[70] F. Wang, S. Liu, P. Liu, Y. Bai. Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams, *EDBT Conference*, 2006.

[71] R. Want. An Introduction to RFID Technology, *IEEE Pervasive*, 2006.

[72] R. Want. Enabling Ubiquitous Sensing with RFID, *Computer*, 37(4), pp. 84–86, 2004.

[73] E. Welbourne, M. Balazinska, G. Borriello, W. Brunette. Challenges for Pervasive RFID-based Infrastructures. *IEEE PerCom Workshop on Pervasive RFID/NFC Technology and Applications (PERTEC'07)*, 2007.

[74] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska, G. Borriello. Building the Internet of Things Using RFID. *IEEE Internet Computing*, 13(3), May–June, 2009.

[75] S. A. Weis, S. Sarma, R. Rivest, D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. *First International Conference on Security in Pervasive Computing*, 2003.

[76] W. White, M. Riedewald, J. Gehrke, A. Demers. What is "next" in event processing? *ACM PODS Conference*, 2007.

[77] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. *2nd CIDR Conference*, 2005.

[78] E. Wu, Y. Diao, S. Rizvi. High-performance complex event processing over streams. *ACM SIGMOD Conference*, 2006.

[79] D. Zimmer, R. Unland, On the semantics of complex events in active database management systems. *ICDE Conference*, 1999.

[80] iAnyWhere Solutions Inc Whitepaper: *Manage Data Successfully with RFID Anywhere Edge Processing*, http://www.sybase.com/files/White_Papers/SybaseRFID_edgepro-053107-wp.pdf.

[81] Hitachi Unveils Smallest RFID Chip. *RFID Journal*, March 14, 2003. http://www.rfidjournal.com/article/articleview/337/1/1.

[82] http://www.boycottbenetton.com

[83] http://www.boycotttesco.com