Privacy-Preserving Event Detection in Pervasive Spaces

Bijit Hore, Jehan Wickramasuriya, Sharad Mehrotra, Nalini Venkatasubramanian, Daniel Massaguer
School of Computer Science University of California Irvine, Irvine, California 92697
Email: {bhore,jwickram,sharad,nalini,dmassagu}@ics.uci.edu

***Abstract*—In this paper, we consider privacy challenges in event-driven pervasive spaces where multimedia streams captured by sensors embedded in the infrastructure are used to detect a variety of application-specific media events. In particular, we develop techniques to detect events without disclosing any identifying information unless necessary. We characterize the nature of inference channels that arise and model privacy preserving event detection as an optimization problem that attempts to balance disclosure with performance. We design and test efficient communication protocols that realize this tradeoff.**

I. Introduction

We consider the privacy challenges in designing human-centric pervasive spaces where *sensors* such as RFID readers, video cameras, biometric devices etc. are used to capture human activities. Consider, for instance, a surveillance that employs sensors to detect unauthorized entry in a region (e.g., by analyzing a video stream [35]). While the video sensor realizes the surveillance objective, it raises potential privacy concerns – e.g., it provides evidence of presence/absence of subjects, authorized or not, to be in the area even if such information is deemed sensitive. Such privacy concerns arise when subjects do not fully trust the pervasive space (e.g., due to fear of misuse of information for tasks other than those explicitly stated, danger of data theft by cyber-thieves and/or insider attacks [1])).

Our goal in this paper is to design event detection systems that meet the stated objectives of the pervasive space (e.g., surveillance) without leaking any additional sensitive information. We first note that if the pervasive space is entirely untrusted, then such privacy preserving event detection will remain elusive. In our approach, we assume tamper-proof sensing devices [37], [36] capable of limited computation that can be programmed to strip identifying information from raw signals [35] (e.g., masking faces) and to generate media specific event streams (e.g., a video camera may sense "entry of a person into a room"). Such an assumption is not unreasonable given advances in sensor technologies (e.g. research on low-cost cryptographic schemes [30], [15], [34]); smart surveillance systems such as IBM's $S3$ [14] that already employ tamper-proof sensors exist.

Privacy preserving event detection in untrusted environments has previously been studied [35], [9], [22] with the focus on developing media specific approaches to obfuscate identifying information. In this paper, we go beyond such simple cases and take on the challenge of privacy-preserving detection of complex events that may be a composition of multiple media-specific events. Composite events can be detected by either storing the incoming sensor data streams in the form of a log and evaluating predicates on it, or by taking an automaton-based approach as in [7]. Either way, the system needs to store past events and/or state information of the automata, which, as we will show later in Section III, could lead to information leakage. Cryptographic schemes (e.g., encrypting the data) and media specific obfuscation methods, while being necessary components of the solution, do not, by themselves, prevent leakage of sensitive information. Information leakage could result from the data access protocols for storage and retrieval of state/event information.

In the remainder of the paper, we adopt an automaton based approach to event detection (described in Sec. II). We assume that sensors detect basic media events and engage in an event detection protocol with an (untrusted) server that maintains all the state information including instantiated (partially executed) automata information. In Sec.III, we characterize the nature of privacy and inference channels in composite event detection. We then design a secure communication protocol between trusted and untrusted components that enables (composite) event detection (Section IV). Since the trusted component is limited in its capabilities (i.e. storage, processing), the challenge lies in designing an efficient, scalable solution that ensures a desired level of privacy while minimizing execution overhead. The trade-off between level of privacy and system performance is modeled as a constrained optimization problem, where the objective is to minimize communication overhead and the constraint is to ensure the required level of privacy. We develop a heuristic for solving this NP-hard optimization problem which gives good results in practice (also described in Section IV). We provide insights into the performance of our protocol and describe an experimental deployment in Section V.

We note that we could have explored privacy preserving event detection using an alternative architecture in which event logs/automata state is stored at the sensors. While this may seem attractive since storage at sensors is trusted, we do not pursue this route due to two reasons. First, from the practical standpoint, detecting composite events may require significant storage of media event for extended period of time which, given the limited storage and computational capabilities of sensors, may not be a scalable solution. Second, since composite events may span multiple sensors, one faces additional complexity of distributed event detection requiring exploration of potential information leakage as a result of communication/interactions between sensors for composite event detection. We relegate such

an exploration to future work.

II. Event Model & System Architecture

We model the pervasive space as a set of users (U), physical regions (P), and resources (R). Sensing in the pervasive space happens through a variety of mediums (e.g., video cameras, RFID, motion detectors etc.). We refer to the raw sensor data gathered by sensors as *media streams*. A media stream S is analyzed to extract *media events*.

Media, Primitive & Composite Events: A *media event* is modeled as a timestamped 4-tuple, $e = (u, s, r, a) : t$ where u corresponds to a user (subject), s a region, r a resource, a the category of the activity, and t designates the time the event occurred. The event e is interpreted as a user u performed activity a, in space s, involving resource r at time t. For instance, a media event (BOB, LOADING_DOCK, *, ENTRY):3:00pm represents that "Bob" entered the "loading dock" at "3:00pm". The resource here was unspecified (*) (could be a null value). The notation $e.u$ will refer to the user associated with event e. In general, media events are domain dependent, with the specification of and mechanisms to detect them being implemented by the designer of the pervasive application. Using such mechanisms, a media stream can be converted into a stream of media events. The set of all media events that can be generated in this space is finite & enumerable they are a subset of the cartesian product of the set of all *users*, *spatial regions*, *resources* and *activities* that can be detected by the sensors. The finiteness of this set has consequences on the privacy-analysis as we will see in Section III and IV.

A *primitive event* refers to a class of media events. A primitive event is also specified using a 4-tuple $< u, s, r, a >$ and is further associated with a temporal condition t_θ, where u refers to a group of (one or more) users, s to the space, r to the resource, a to the type of activity. θ is a simple operator of the form {+,-}, which indicates before ($-$) or after ($+$) time t. For instance, $<$ $u \in$ STUDENT, $s \in$ CS II, $\star$ ENTRY$>$:15:00$_+$ is an example of a primitive event[1]. A media event is said to *match* a primitive event, if a media event *instantiates* the primitive event. Therefore, a media event (BOB, ROOM 113, BRIEFCASE, ENTRY):16:46 matches the aforementioned primitive event, if $BOB \in STUDENT$, $ROOM$ 113 $\in CS\ II$ and the time of entry is after 3:00PM. A result of the match is a "binding" of the variables u, s, r in the primitive event by the corresponding individual "Bob", resource entity "briefcase" and space entity "Room 113" specified in the media event.

A *composite event* is a combination of one or more primitive events. We restrict composite events to those that can be expressed as regular expressions over primitive events. Regular expressions, while limited in expressibility, have been deemed to be sufficiently powerful for a large class of pattern and event detection systems [26], [7].

Composite Event Detection Composite events are translated into their corresponding finite state automata (FSA) referred to as an *event automaton*. An event automaton is a directed multigraph, where nodes correspond to states and edges to state transitions. Edges (transitions) are adorned with a corresponding primitive event that cause the state transitions. The automaton F executes as follows. F, when initiated is in the initial state s_0. At any state s_i, F has a set of possible transitions t_j each of which is associated with a corresponding primitive event p_j. Let $p_j = (u_p, s_p, r_p, a_p) : t_p$, and $e = (u_e, s_e, r_e, a_e) : t_e$ be a media event that *matches* the primitive event p_j. Such a media event will cause (1) F to transition from state s_i along the transition t_j, and (2) bind the variables associated with the primitive events based on the matching media event. The following example illustrates how an automaton is used to capture a composite event of interest for inventory tracking purposes.

Example: Inventory Tracking is done on two floors of a building to determine if any member of the *database* research group enters either of these floors and then leaves with a projector. The corresponding automata can change states on the following primitive events. Note that $\overline{u}$ refers to the instantiated value of u.

$e_1 \equiv <$ $u \in$ DATABASE, FLOOR_2, *, ENTRY$>$
$e_2 \equiv <$ $u \in$ DATABASE, FLOOR_4, *, ENTRY$>$
$e_3 \equiv <$ $\overline{u}$, FLOOR_2, *, EXIT$>$
$e_4 \equiv <$ $\overline{u}$, FLOOR_4, *, EXIT$>$
e_5 = $e_6 \equiv <$ $\overline{u}$, STORAGE_ROOM, PROJECTOR, ACCESS$>$

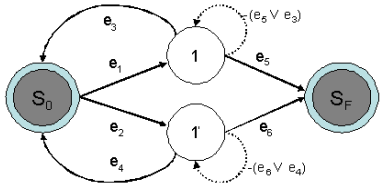

Fig. 1. Automaton for inventory tracking.

In our automata, for every state there is always a transition for all possible media events. We differentiate between edges of the automata that are used to drive the automata to its successor state (α edges) from the edges that are self-loops used to filter out media events unrelated to the progression of the automata (β edges). A similar concept of filter edges was used in [7], which consume events that do not drive a particular automaton to advancing its current state. Note that while implementing automata of this kind, we can effectively ignore β edges as they do not alter the state of the automata. We will henceforth ignore β edges of automata. The automata are *individual-centric*. That is, for any sequence of media events $e_1, e_2, \ldots, e_n$ that transitions an automaton F from the initial state s_0 to its final state s_F (using only the α edges), $e_1.u = e_2.u \cdots = e_n.u$, where $e_i.u$ refers to the user associated with the event e_i.

In the remainder of the paper we will require the following notation. For an automaton F, we define the notion of $USER(F)$. $USER(F) \subseteq U$, is the subset of individuals such that $x \in USER(F)$ if and only if there exists a sequence of media events $E = e_1, e_2, \ldots, e_n$, $e_i = (x, s_i, r_i, a_i) : t$ that can be recognized by F (that is, they can transition F from its initial state to a final state). For instance, in the example above all members of the DATABASE group would be in the $USER(F)$, for the automaton depicted in Fig. 1. In contrast, "Bob" would not be in $USER(F)$ if "Bob" was not a member of the

[1]We will, for notational simplicity, sometimes ignore temporal constraints in the specification of primitive events if the temporal constraints are not integral to the concept being discussed.

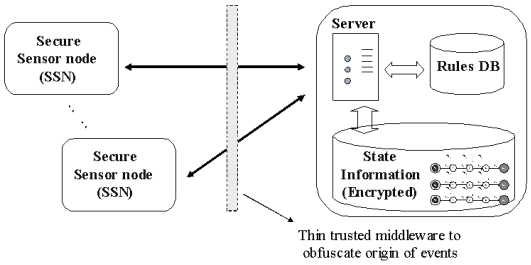

Fig. 2. System Architecture

database group. We generalize the concept of $USER$ to the set of automata F_S by defining $USER(F_S) = \bigcup_i USER(F_i)$, $F_i \in F_S$. An automaton F is associated with a user u if $u \in USER(F)$.

System Architecture: Our event detection system consists of multiple components (depicted in Fig. 2). The two primary components of interest are the secure sensor nodes (SSN) and the untrusted server with large storage and computational capacity that stores state information about the pervasive space.

SSNs: The secure sensor nodes (SSNs) are also the media event generators that convert the media stream into sequence of corresponding media events. The SSNs consist of one or more sensors, limited storage, and computational resources. For example, a SSN may consist of a video camera attached to a processor and storage that can do some limited processing on the stream [35], [14] (e.g., image processing computations). It could also consist of a RFID reader which is responsible for detecting a set of RFID tags. We require the presence of a trusted (thin) middleware that is able to obfuscate the origin of events. We will currently assume that media event generation can be performed without revealing any potentially harmful information to the server. (This assumption may not hold in general – we discuss its implications as well as methods to resolve the resulting privacy challenges in the extended version of the paper [13]). The SSNs are trusted and assumed to be tamper proof. Encryption (decryption) can only be performed within the secure perimeter of an SSN using a symmetric key encryption scheme (like DES). The encryption keys are also stored exclusively within the secure perimeters.

Server: A central server (referred to simply as a server) stores the automaton objects. The server also communicates with all the SSNs deployed in the space using the secure protocol (to be described later) to update state information.

We denote an instantiation of our event detection system by $\mathfrak{S}$, which comprises the following two components discussed below: (i) a data model & storage scheme ($\mathfrak{D}$); (ii) a suite of communication protocols ($\mathfrak{P}$).

Data component $\mathfrak{D}$: The state information (automata) are always stored in encrypted form on the server in a table where each row is mapped to a distinct automaton. Each automaton corresponds to a "rule-individual" pair, where rule denotes a composite event, for example, "*(5 cups of coffee in a day,* Tom*)*", "*(Server-room entry with trolley after 6:00 pm,* BOB*)*" etc. The size of the table is constant i.e., total number of automata in the state-table is fixed, which we denote by NUM_RULES. Each encrypted automaton is tagged by a *label* that is used only for lookup purposes. The tag is used only for set-membership queries and does not divulge any information beyond membership[2].

Communication protocol $\mathfrak{P}$: Fig. 3 shows the generic template of the communication protocol between a SSN and the server in order to service a media event generated by the SSN. Since SSN is the only secure (trusted) component, all communication data between the server and a SSN needs to be encrypted. Following are the sequential steps in an event-servicing protocol initiated by the SSN:

- On event e, the SSN generates a message $M(e)$ that uniquely identifies the event and encrypts and sends them to the server. The ciphertext is denoted by $E_k(M(e))$.
- The server on receipt of the message, returns the set of encrypted automata to the SSN that are tagged by $E_k(M(e))$. (This can be implemented using a cryptographically secure keyword matching scheme similar to those proposed in [31]).
- The SSN decrypts the set of automata one at a time and advances the state of each automaton if necessary.
- The SSN re-encrypts the automata (non-deterministically[3]) and sends them back to the server. If any automaton reaches its final state, the SSN notifies the server of event detection.

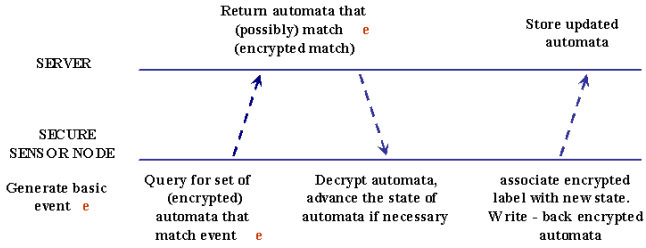

Fig. 3. A generic protocol to service events.

We will refer to the above description of the system and protocol as the *reference implementation* from here onwards.

III. Privacy Requirements

In an event-processing system such as ours, the knowledge that an individual is associated with an event is considered sensitive. Our goal is to disallow association of event to an individual. So complete privacy is achieved when an event could be potentially attributed to any of the N individuals associated with the system. Of course, one cannot ensure this as long as there is external information. For instance, say an event occurred and if the adversary knew that JOHN was absent on the particular day, then JOHN is eliminated from the set of possible users, thereby violating N-anonymity. We make our goal a bit simpler. We consider that adversary knows only the encrypted event logs and also has complete knowledge of the environment (i.e., of all the automata). His goal is to determine from the event log which event corresponds to which individual. We will show

[2]The notion of set membership will become clear in Section IV

[3]non-deterministic encryption ensures that multiple encryption of the same plaintext generate different ciphertexts.

complete privacy (N-anonymity) can be achieved very simply but at a very high cost, and therefore we will explore a way to tradeoff privacy with performance. Specifically, we develop and algorithm to achieve k-anonymity where $k \leq N$. We will exploit this relaxation to gain efficiency. Formally, the k-anonymity criteria for event-processing systems is the following:

Criteria 1: **(k-anonymity Criteria)** Given a media stream S and a sequence of corresponding media events with timestamps $\{e_1{:}t_1, ..., e_n{:}t_n\}$, a solution is k-anonymous if it ensures that $\forall i \in [1, n]$, $e_i.USER$ **cannot be mapped** with certainty to a set of less than k individuals at any $t_j, j \geq i$ (i.e., at any time instant after e_i is generated).

Note that, the above criteria ensures that each individual is *indistinguishable* from at least k-1 others at all times. We note that the above concept of privacy (i.e., $|USER(e)| \geq k$) is similar in spirit to the concept of k-anonymity in data publishing [25], [32] where k-anonymous dataset implies that each record could be associated with k or more individuals. It was soon realized that simply the notion of k-anonymity is not good enough in a publishing scenario since the sensitive attribute of all members of an anonymity set might have the same value (say "disease = AIDS"). This led to more stringent criteria such as l-diversity [3], t-closeness [17] etc. to be proposed. However, k-anonymity works in our case for 2 main reasons: (i) There is no specific sensitive field (like "disease"); (ii) Unlike in data publishing where the adversary may knows whether a record corresponding to a person is in the database or not, in our case, such knowledge is not there. For instance, there may be no event corresponding to JOHN at time t_1 even though JOHN is in the anonymity group of TOM who is associated with an event at t_1. As a result, inferences of the kind that occur in data publishing do not occur in the our setting.

Security & Adversarial Model: The key challenge we address is this paper is that the server-side environment is untrusted. In real life, there maybe one or more malicious insiders on the server-side who can be regarded as adversaries e.g., database administrators. The goal of the adversary is to deduce the identity of the individuals involved in every media event that is generated by the SSNs. We will assume a *passive adversary* i.e., the adversary is only interested in gleaning information about the events, and does not disrupt the normal functioning of the system in any manner.

We will assume the following background knowledge is available to the adversary: (i) the set of media events that can be generated by the sensors; (ii) all the N individuals that interact with the space; (iii) the rules-to-individuals mapping (i.e., which rules apply to which individuals); (iv) the details of the automata that implement these rules.

Now, we take a closer look at the nature of inference and formally define the inference mechanism that can be employed by an adversary described above.

A. Inference channels

It is easy to see that a simple "scrubbing" of the data does not prevent disclosure from an adversary who has the above mentioned background knowledge. For instance, if the adversary sees the following scrubbed representation of an event – "<X, SERVER-ROOM, *, ENTRY>: 8:35 pm" and also knows that only TOM has access to the server-room, he can easily infer the identity of the individual X.

Information about a particular event e^* may be indirectly inferred by observing the effects of other events before and after e^*. In particular the adversary may be able to identify the individual involved in e^* by combining the observed automaton access patterns and the background information that he has as illustrated in the following example.

Example: Let there be two rules F_1 and F_2 such that $USER(F_1) = \{I_1, I_2\}$ and $USER(F_2) = \{I_2, I_3\}$. On a media event e, let the SSN execute the protocol of Fig. 3. In the second step of the protocol, if the SSN retrieves exactly one row from the server, it could be an automaton belonging to either one of the 3 individuals and therefore 3-anonymity is guaranteed. However, if it retrieves two rows, the adversary is able to instantly deduce that the individual involved (i.e., $e.USER$) could only have been I_2, leading to privacy violation. $\diamondsuit$

Inference channels are observable features of $\mathfrak{D}$ or $\mathfrak{P}$ or a combination of both on a sequence of events generated in the space. Inference channels exist due to the very nature of the rules (composite events) that are defined in the space. For instance, if all rules applied to all N individuals then any solution that does not explicitly reveal the contents of the state table and non-deterministically encrypts all the messages exchanged between SSN and the server, is able to guarantee N-anonymity. The differential nature of the rule set, i.e., the fact that "different set of rules can apply to different individuals" makes inferencing possible. As illustrated by the previous example, the important point to note is that encryption by itself is not enough to obfuscate the access patterns of automata which can give away enough information to uniquely link an event to an individual. Now, we formally characterize how access patterns allow inferencing and present an approach to obfuscate these patterns.

An observed *access pattern* (simply referred to as a *pattern* for short) is formally defined as follows.

Definition *1:* **(Pattern)** A pattern is any sequence of sets of literals, where a literal denotes an automaton-id. A pattern $p_{n,m}$ denotes a sequence of n sets, each with at most m literals.

A pattern denotes a sequence of row/automaton accesses on consecutive events. For example if $F^{'}$, $F^{''}$ and $F^{'''}$ are 3 automata, a pattern $p_{4,2}$ could be the following. $\{\{F^{'}, F^{''}\}, \{F^{'}\}, \{F^{'}, F^{'''}\}, \{F^{''}\}\}$.

A *pattern template* is defined as follows.

Definition *2:* **(Pattern-template)** The template of a pattern denotes the generic class to which the given pattern belongs. It is denoted by replacing the actual literals in the pattern by a variable.

For example, the above pattern follows the template $\{\{x_1, x_2\}, \{x_1\}, \{x_1, x_3\}, \{x_2\}\}$ where $x_1 \leftarrow F^{'}$, $x_2 \leftarrow F^{''}$ and $x_3 \leftarrow F^{'''}$ (We assume that a suitable sequence of events exists which generates this pattern of automaton access). In general there maybe multiple patterns following the same

template. A *characteristic pattern* of an automaton is defined as follows:

Definition *3:* **(Characteristic Pattern)** A characteristic pattern of length n for an automaton F is an instance of a pattern-template p (of length n) where F is present in each of the n sets in the pattern.

For example the pattern $\{\{F^*, F^{'}\}, \{F^*, F^{''}\}, \{F^*, F^{'}, F^{''}\}\}$ is a characteristic pattern of the automaton F^*. Each automaton is associated with a (possibly infinite) set of such characteristic patterns depending on its structure. The *set of all characteristic patterns* of each automaton might be unique. We will use the term "characteristic set" of an automaton F to refer to the set of all characteristic patterns of F and denote it by $\mathbf{CP(F)}$.

Example: The figure below shows 3 automata (denoted x, y and z for short) corresponding to 3 rules applicable to an individual TOM. The figure also shows some of the characteristic patterns of these automata. In general the characteristic set of an automaton (as well as some subsets of it) could be unique.$\diamondsuit$

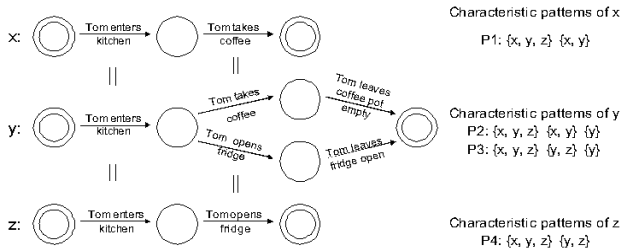

Fig. 4. Characteristic patterns.

Pattern analysis by the adversary: An adversary with a very large storage and computational power can be expected to know (determine) all characteristic patterns (say, of all lengths up to some large n) for each automaton. As a result, after observing sufficiently long sequence of events and the corresponding row accesses he is able to match[4] rows in the table to specific automata and thereby determine the identity of the individuals. This leads to a definition of observable indistinguishability between two automata. For a given a media event e, the adversary can only infer the identity of $e.USER$ indirectly by determining the set of automata that were affected while servicing e. In our scheme since the automata are encrypted, they may only be identified by recognizing the characteristic access patterns.

Observable indistinguishability: Two automata are said to be *observably indistinguishable* or simply indistinguishable from each other if their characteristic sets are *identical*. We can now define *pattern isomorphism* between two sets of automata as follows.

Definition *4:* **(Pattern Isomorphism)** Let $\mathcal{A}$ and $\mathcal{A}^{'}$ be 2 sets of automata where $|\mathcal{A}| = |\mathcal{A}^{'}|$ and $G : \mathcal{A} \rightarrow \mathcal{A}^{'}$ be a bijective (i.e., 1-1 and onto) map from $\mathcal{A}$ to $\mathcal{A}^{'}$. Then G is called a pattern isomorphism iff $\forall$ automaton $a \in \mathcal{A}$ and $G(a) \in \mathcal{A}^{'}$, there is a natural bijection $T_{(a,G)} : CP(a) \rightarrow CP(G(a))$ where (i) pattern $p \in CP(a)$ and $T_{(a,G)}(p) \in CP(G(a))$ are instances of the same template, $template(p)$ and (ii) $\forall$ variables x_i appearing in $template(p)$, if $x_i \leftarrow t$ in p then $x_i \leftarrow G(t)$ in $T_{(a,G)}(p)$ where $t \in \mathcal{A}$ and $G(t) \in \mathcal{A}^{'}$.

Under a given isomorphic map G between two sets of automata, the pre-image and image automaton are observably indistinguishable from each other. Now if such an *automorphism* exists for $\mathcal{A}$ (i.e., pattern-isomorphism from the set onto itself), then each automaton and its image under this map will be observably indistinguishable to the adversary. Let $G^* : \mathcal{A} \rightarrow \mathcal{A}$ be a *fixed-point free* automorphism (i.e., $a \neq G^*(a), \forall a \in \mathcal{A}$) such that $a.USER \neq G^*(a).USER \ \forall a \in \mathcal{A}$. If such an automorphism exists, then each automaton pertaining to any individual is observably indistinguishable from some automaton belonging to another individual. As a result, the true identity of the individual associated with any automaton can never be uniquely determined by simply observing the access pattern of an automaton. We will call such automorphism (if it exists), a *non-identifying automorphism*.

We now state the necessary and sufficient condition for achieving k-anonymity. We use the following definition of *diversity* of an automata set:

Definition *5:* **Diversity:** The diversity of a set of automata is the distinct number of individuals represented in the set.

Theorem *1:* **(k-anonymity: Necessary & Sufficient condition):** Given an injective map (assignment) $G_0 : \mathcal{A} \rightarrow R$ from set of automata to the set of rows in the state-table, a sequence of n media events $E_0 = \{E_0(1), \ldots, E_0(n)\}$ $(n \rightarrow \infty)$ and the corresponding row-access pattern $p(E_0, G_0)$, the solution scheme $\mathfrak{S} = (\mathfrak{D}, \mathfrak{P})$ is k-anonymous iff there are at least k-1 other sequences of n events $E_1, \ldots, E_{k-1}$ and corresponding automaton-to-row map $G_1, \ldots, G_{k-1}$ such that (I) patterns $p(E_0, G_0) \equiv p(E_1, G_1) \equiv \ldots \equiv p(E_{k-1}, G_{k-1})$ and (II) $E_0(i).USER \neq E_1(i).USER \neq \ldots \neq E_{k-1}(i).USER$, $\forall i = 1, \ldots, n$.

As it turns out, the problem of detecting whether a given set of automata is k-anonymous is a difficult one. We show that the case for $k = 2$ is NP-Complete by reducing the known NP-complete problem that decides "Whether a graph has a fixed-point free automorphism or not" [18] to an instance of our problem. We conjecture that the problem is NP-complete for higher values of k as well. The problem now is to come up with an automaton annotation scheme that on one hand guarantees that correct automata are retrieved on each event and on the other ensures that the access patterns of the automata are k-anonymous. In the next section, we provide solution that meet these constraints. The proofs of theorem 1 and the reduction is given in [13].

IV. ANONYMITY VIA PATTERN OBFUSCATION

In the previous section, we saw that it is difficult to determine whether the patterns generated by an arbitrary set of automata satisfy the k-anonymity criteria. This forces us to consider only a restricted class of automaton annotation schemes where the set of observable access patterns are provably k-anonymous. Consider the following scheme:

[4]Matching can be done by comparing the characteristic pattern of the row and those of the automata which he can pre-compute

Solution 1: **(k-Individuals Partitioning)** Let $M = \lfloor N/k \rfloor$ where N denotes the total number of individuals. Make M disjoint bins of individuals, where each bin has at least k individuals. Tag all the media events appearing in any automaton within a bin with the "bin-id".

Note, that the above scheme assigns a static label to each automaton in the state-table (i.e., the index-tag does not reflect the change of state). The SSN requests the server simultaneously for all automata assigned to a bin which leads to a homogenous access pattern for all of these automata, hence making them indistinguishable from each other. As a result, this solution satisfies our k-anonymity criteria. Next, we show that solution 1 can be significantly improved upon in terms of performance. The improved solution is based on the notion of a *connected group* which we define below.

Definition *6:* **(Connected Group)** The connected groups of automata for an individual I correspond to the set of connected components in an undirected graph $G_I = (V_I, E_I)$, where V_I has one vertex corresponding to each automaton of I denoted F^I and there is an edge in E_I corresponding to every pair (F^I_x, F^I_y) where both automata have at least one common transition (α edge) between states.

Example: Fig. 5 shows two connected components corresponding to TOM. All the 3 automata in the first group can make a transition on the event "TOM, KITCHEN, *, ENTRY" and the second component corresponds to the automata with the common edge "TOM, SERVER_ROOM, *, ENTRY".

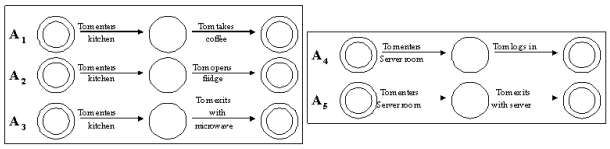

Fig. 5. Example: Connected Groups

As illustrated in the example, a connected-group always represents a single individual, but there may be multiple connected-groups corresponding to an individual. Now, the second more efficient solution is outlined below.

Solution 2: **(k-connected-group Partitioning)** Partition the connected-groups of automata into bins such that each bin has diversity $\geq$ k (i.e., has automata representing at least k different individuals). Tag all the media events appearing in any automaton within a bin with the "bin-id".

It is easy to see that the above scheme too guarantees k-anonymity. Thus, the above partitioning scheme may assign two connected groups of automata corresponding to the same individual to two distinct bins[5]. This added flexibility increases the size of the solution space as compared to the more restrictive scheme proposed in Solution 1, and in general leads to lower cost (more efficient) solutions. The modified communication protocol (for both, Solution 1 and 2) is depicted in figure 6.

[5]This in effect allows multiple *avatars* of a single individual to exist as if they were completely different individuals

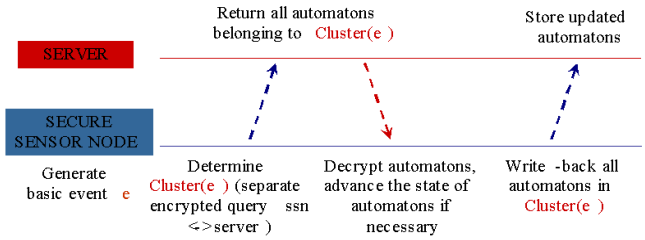

Fig. 6. Secure protocol for servicing events.

A. Minimizing Partitioning Cost

The solutions proposed above poses a natural optimization problem: since different automata may be accessed at differing frequencies, some partitioning schemes will have lower average *costs* (i.e., number of false-positive retrievals) than others. The goal is to partition connected groups of automata so as to minimize this cost while ensuring at least k different individuals are represented in each bin. (Observe that, Solution 1 is a special case of Solution 2, therefore, we will only discuss the solution for the second scheme).

We now model the above problem as a novel set partitioning problem with some unique constraints and show that it is NP-hard. Subsequently, we propose a heuristic algorithm that generates good (low cost) partitions in practice.

Modeling as a "minimum-cost partitioning" problem: Let us assign each of the N individuals in the system a distinct *color*. Let each automaton belonging to an individual be represented by a ball. Each ball has 3 attributes: *color, price* and *weight*. The color of a ball is same as that assigned to the individual it corresponds to. Initially, each ball has unit price and a weight equal to the number of edges in the corresponding automaton. The price of a ball represents the space that the corresponding automaton takes up in memory (and consequently the transmission overhead in the communication protocol). The price is set to 1 (unit cost) initially since each automaton occupies a single row in the state table. The weight is a measure of how often an automaton is accessed in the environment. We set the weight equal to the number of edges under the assumption that number of edges is proportional to the probability with which an automaton is accessed on a randomly generated media-event. An alternative could be to use the number of distinct media events that can cause a state-transition in the automaton. In any case it does not affect the partitioning algorithm.

In the pre-processing phase, the connected components are computed as follows: all automata of the same color that have at least one edge in common are fused into a new "larger" ball of same color. The weight and price of the fused ball are set equal to the sum of the corresponding values of the component balls (i.e., the balls that were fused). This is correct since both the measures are completely additive under our assumptions and the system design i.e., the probability of accessing a connected group of automata is equal to the sum of the individual access-probabilities of the component automata, as is the transmission overhead. Now, the optimization problem can be stated as follows.

Problem Statement 1: **(Optimal k-anonymization)** Partition

the set of colored balls into bins (allowing as many bins as required) such that the number of distinct colors represented in each bin is at least k and the cost of the binning strategy ($\mathfrak{S}$) is minimized.

In the k-anonymous protocol since all rows within a bin are to be accessed together, the associated *cost* of a solution scheme $\mathfrak{S}$ is given by the following expression.

$$Cost(\mathfrak{S}) = \sum_{B \in Bins} (\sum_{ball \in B} weight(ball)) \times (\sum_{ball \in B} price(ball))$$

The complexity of the optimization problem: The above optimization problem turns out to be NP-complete in the general case. The "minimum sum of squares" problem [10] which is a known NP-hard problem, can be reduced to an instance of our balls-and-bins optimization problem. (Details are relegated to the longer version of the paper). We also note that, the more constrained case, where all automata of an individual are forced to be in the same cluster, is also a NP-complete optimization problem as can be shown by a similar reduction.

B. A Heuristic for Least-Cost Binning

We give a simple heuristic solution to the above optimization problem. In practice the algorithm leads to good solutions. In this paper we make no attempt to give a theoretical guarantee on the approximation factor. Instead we provide an intuition to the reader about the nature of performance scale-up that can be expected through empirical results using a test set of events, automata and individuals.

The input to the problem are 3 arrays of size equal to the number of balls $|B|$. The first array $Color[1 \ldots |B|]$ specifies the color of each ball, the 2^{nd} array $Weight[1 \ldots |B|]$ specifies the weight of a ball and the last array $Price[1 \ldots |B|]$ specifies the price of each ball. We also specify the number N of distinct individuals (colors) in the system and the required anonymity level k. The output is the cost of the binning scheme and an array $Bins$ (of size $|B|$), which specifies the id of the cluster assigned to each ball. When the anonymity constraint is k, it is easy to see that the number of distinct clusters (bins) cannot be more than $\lfloor |B|/k \rfloor$. The algorithm first starts with a randomly generated feasible solution and then iteratively decreases the solution cost by carrying out cost-reducing "ball transfers" and "ball exchanges" between bins till no more such transfers and/or exchanges are possible. The detailed pseudo-code is presented in Algorithm 1.

In the algorithm, the candidate ball-transfers and their associated cost reductions can be modeled as a directed graph with the nodes corresponding to bins and edges having integral weights. A (directed) edge from a node n_1 to n_2 denotes the best candidate ball-transfer from the bin corresponding to n_1 to bin corresponding to n_2. The edge weight denotes the net cost reduction due to the corresponding transfer which assume that the source and target bins are not involved in any other transfers. A feasible ball transfer refers to one which does not violate the k-anonymity constraint, that is, each bin should have at least k colors represented at all times. A linear-time graph matching algorithm (like [33]) can be utilized for computing the **most profitable** matching in the graph. Ball-exchanges[6] between bins can also be modeled similarly. Finally, since the ball transfers are only carried out for positive cost-reductions, Algorithm 1 always terminates.

Algorithm 1 Least cost k-diverse partition.

```
Input:Color[1, ..., |B|],Weight[1, ..., |B|],Price[1, ..., |B|], k, NUM_COLORS;
Output: Cost, Bin[1, ..., |B|];
if k > NUM_COLORS then
    Print: "No solution possible for given k";
    Return φ;
end if
/* Generate an initial feasible solution */
X ← set of all balls; i ← 0;
while k or more distinct colored balls in X do
    Initialize new bin B_i;
    Randomly put k distinct colored balls from X to B_i;
    i ← i + 1; X ← X \ B_i;
end while
Let M ← i; /* num bins initialized */
if M = 1 then
    for i = 1 to |B| do
        Bins[i] = 1;
    end for
    cost ← (Σ_{i=1}^{|B|} Weight[i]) × (Σ_{i=1}^{|B|} Price[i]);
    Return (cost,Bins[1 ... |B|]);
end if
if ∃ balls not assigned to any bin then
    Assign each such ball to a random bin B_i, i ∈ [1, M];
end if
/* Iteratively carry out cost-reducing transfers & exchanges */
while 1 do
    C ← φ; /* set of candidate transfers */
    for all i, j where 1 ≤ i, j ≤ M and i ≠ j do
        C ← C ∪ best feasible ball transfer from B_i to B_j with a positive cost reduction;
    end for
    if |C| ≥ 1 then
        P ← most profitable set of compatible transfers in C;
        Execute all transfers in P;
        Reflect new bin assignments in Bins[1 ... |B|];
        Compute the new cost of partitions;
    end if
    C' ← φ; /* set of candidate exchanges */
    for all i, j where 1 ≤ i, j ≤ M and i ≠ j do
        C' ← C' ∪ best feasible ball exchange between B_i and B_j with a positive cost reduction;
    end for
    if |C'| ≥ 1 then
        P' ← most profitable set of exchanges in C';
        Execute all exchanges in P';
        Reflect new bin assignments in Bins[1 ... |B|];
        Compute the new cost of partitions;
    end if
    if |C| == 0 AND |C'| == 0 then
        /* No candidate transfers or exchanges */
        Return (cost, Bins[1, ..., |B|]);
    end if
end while
```

V. Implementation & Evaluation

We built a prototype of the system described in this paper on top of the SATware-Responsphere framework [12], [2]. SATware is a middleware framework for programming and testing pervasive space applications. SATware is deployed on top of Responsphere, which is a large communications, storage, computing, and sensing infrastructure that covers almost a third of UCI campus. It includes more than 200 sensors (including cameras, RFID readers, temperature sensors, acoustic sensors,

[6]Again, this operation is carried out only if it does not violate the k-diversity constraint and leads to reduction in the cost.

Fig. 7. The user's identity is concealed when he has his 1^{st} (a) and 2^{nd} cup of coffee (b), but revealed when he has his 3^{rd} cup (c).

gas sensors, accelerometers and people-counters), different communication technologies (such as Ethernet, Wi-Fi, Power-line, and IEEE 802.15.4), and several storage and computing servers.

A. Evaluation

We modeled a real-world application based on some actual observed activity on our floor in the office building. We defined 4 groups of people STUDENT, FACULTY, STAFF, VISITOR with 300 members[7] in all and defined 15 rules over these groups. The instrumented part of the floor consisted of the following 3 rooms, KITCHEN, SERVER-ROOM, FACILITIES-ROOM where events could be detected using sensors. The 15 rules were categorized into three sets of 5 each corresponding to each of the 3 rooms. The rules belonged to either of the two categories; (a) Protection of resources and (b) Suspicious activity. These rules were very similar in flavor to those presented in Section II. We then constructed the corresponding finite-state automata that would implement these rules (i.e., detect the corresponding composite events). All 5 automata within a rule-set had at least one event in common and therefore induced a single connected group at most. For e.g., each kitchen-rule fired on an "entry into the kitchen" event. As a result, all the automata pertaining to the kitchen rules that were applicable to an individual would fire when the individual enters the kitchen. None of the rules belonging to different sets had any event in common, for e.g., no automaton implementing SERVER-ROOM or FACILITIES-ROOM rules are affected by any event that takes place within the kitchen and like-wise for the other two sets of rules. (As a result, there are at most three connected groups corresponding to each individual.). Figure 5 shows a few automata corresponding to TOM. All the 15 rules and their corresponding automata specifications can be found in the extended version of this paper.

Through the implementation and its execution in a real-world deployment, we were able to validate the functioning of the algorithm and system.The actual prototype addressed several implementation level issues such as event-detection at runtime, key management, concurrent updates, etc. Some details about these aspects can be found in the longer version of the paper [13]. Figure 7 illustrates a sample scenario – a series of snapshots of different states associated with composite event-detection as implemented in our system. The camera (which is the SSN in this case) reveals an individual's identity only when he/she consumes more than 2 cups of coffee.

Performance of partitioning algorithm: To further study the performance and security/scalability tradeoffs of the partitioning algorithms however, we emulated event sequences based on the above dataset. We compare the performance of two solutions schemes: (i) *k-Individuals partitioning* and (ii) *k-connected-group partitioning*. We use the algorithm described in the previous section to compute the optimal solution in both cases. We then compare the performance characteristics predicted by the algorithm (i.e., cost estimates) with the actual number of automata retrieved in servicing a sequence of randomly generated 1000 events.

To generate the emulated event sequence, we associate a weight with each automaton that reflects the frequency with which it is accessed in the application. While in the real world, the weight would be some function of the frequency with which a set of specific events occur and how they are inter-related, we simplify this mapping by associating a random positive (small) integer as the weight of each automaton and set the net weight of each ball (connected group) as the sum of the weights of automata belonging to that connected group. From this, we generated an event-sequence where each event was simply denoted by the id of a ball picked with a probability proportional to its weight.

Figure 8 shows the **estimated costs** (given in previous section) of the 2 partitioning schemes for various values of k (the required anonymity level) and compares them with the naive algorithm. The dataset we generated had 300 people, 772 balls in the unconstrained case and 300 balls in the constrained case (since all automata of an individual are forced to be in the same bin). The graph shows that the cost differential is expected to increase consistently with increasing value of k.

For the second plot, we generated a sequence of 1000 media events as described above. Figure 9 shows load characteristics of the two clustering schemes on this test sequence. The y-axis is the total number of automata retrieved over the 1000 events which represents the transmission-load due to the "SSN-server communication" in a real setting.

The two important conclusions that can be drawn from observing the two plots are that: (a) The simulated transmission overhead in Figure 9 varies (grows with k) in almost an identical manner to what is predicted by the partitioning algorithm (cost estimate in Figure 8) and (b) With increasing k, the load on the system increases at a much faster rate for the *k-Individuals partitioning* based anonymization approach as compared to the *k-connected-group partitioning* one. While the simulation result displays a very high degree of conformance with the predicted results, we expect the load characteristics to diverge from the model in a real-life setting. The reason being that the weight of an automaton is always going to be an approximation of the real value and can never be estimated with complete accuracy.

Evaluation with different rule sets: The above experiments were conducted in the context of the pervasive space deployment discussed earlier with a limited number of rules and hence automata. To understand the performance of the partitioning technique under varying rule conditions, we performed other experiments using synthetic data that was generated by varying various parameters such as average "connectivity" between rules, number of individuals, and number of distinct rules.

The synthetic dataset generated involved a set of rules mod-

[7]There were 48 individuals on the floor, but for our simulations we artificially increase the number to 300, keeping the proportions of each group approximately the same.

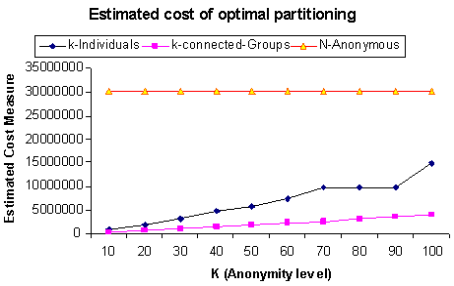

Fig. 8. Cost estimates of constrained ("Together") & unconstrained ("Separate") partitioning schemes

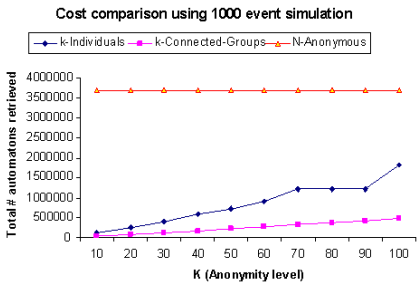

Fig. 9. Total # automata retrieved in 1000 events

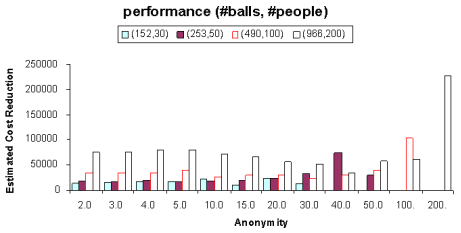

Fig. 10. Analytical estimate of performance improvement

eled using a random graph $G = (V, E)$. V is the set of vertices which represent the set of distinct rules applicable in the pervasive space. An edge between any two nodes is added with a predefined probability *connectedness*. An edge $e = (u, v) \in E$ represents the fact that two rules u and v are connected. Each vertex has two associated metrics described previously: *price* (set to 1) and *weight* (a random small positive integer). To generate the set of rules applicable to an individual x we create a new graph G_x by considering each individual x in the system at a time and selecting a subset of vertices $S_x \in V$ as being the rules that apply to x. Edges are retained if both endpoints belong to S_x. All vertices in S_x are assigned the color c_x (which is unique for each individual). Thus, the connected components of G_x denote the connected groups for individual x. These connected nodes are fused into a *super-node* (i.e., a ball in our terminology) such that the weight (price) of the new super-node in G_x is the sum of the weights (prices) of its constituent vertices. We generate such sets of balls, each set having a new color corresponding to each individual in the pervasive space. These balls along with their color, weight and price comprise the input to our clustering algorithm.

Fig. 10 shows the **estimated** *absolute cost difference* between the 2 solution schemes for various values of the parameters: #_RULES, #_PEOPLE, ANONYMITY and AVG_NUM_RULES. The # BALLS denotes the total number of connected groups generated from the set of all automata in the state-table. The plot shows that the cost differential is expected to increase consistently with increasing number of individuals and automata in the state-table, but shows no clear trend across varying values of k.

VI. RELATED WORK

There is a large body of work in the area of systems for pervasive computing environments [20], [29]. GAIA [28] is an example of a system designed inherently to enable pervasive computing. Other systems such as CRICKET [27], tackled the problem of localization for embedded sensor and mobile context-aware applications.

The importance of privacy to pervasive computing environments for system designers and users has been raised in [6], [16]. Kandur et. al [16] state that even in a resource-constrained, distributed environment the need to provide security and privacy assurances are still paramount. The particular environment under examination is a infrastructure where small, dispersed video-audio sensors feed data to a central observation station which is not dissimilar to that being presented here. The difference is that they examine lower network level security and cryptography solutions for sensor networks, whereas our approach is a more general technique for event detection. Langheinrich [19] stated the danger current pervasive systems research faces in continuing on without considering privacy as an inherent part of system design.

Research on privacy for pervasive computing environments has been looked at from multiple angles. For example, the MIST system [4] combines hop-to-hop routing based on handles with limited public-key cryptography to preserve privacy from eavesdroppers and traffic analyzers. Our previous work [35] sought to address issues in preserving the privacy of users in the context of real-time video surveillance. Some recent work on event otologies for video-based events (VERL) [21] comes very close to the type of event detection we envision for pervasive spaces. It allows one to capture various basic and composite events using a formal language. Our implementation uses automata for representing composite events due to the natural fit with the type of activities of interest in a pervasive space. More recently, Oleshchuk [23] has investigated privacy-preserving monitoring and surveillance, which is in the domain that some of our scenarios are presented. The approach taken here however is to utilize multi-party computation and pattern-matching in event sequences. The difference is that here the focus is very specifically on pattern-matching, where the event data is measurement data from sensors and the privacy criteria is whether to disclose data from a particular sensor or not. Our automaton-based approach is more general and could be utilized to realize such an application as well. Here cryptographic techniques are utilized to do event detection based on pattern-matching over infinite alphabets.

Other types of event-based systems [11], [21] have implementations based on Petri Nets, which support concurrent behavior and parametrization. However, even for simple expressions, they quickly become complicated and expensive to implement. Other recent work in ubiquitous computing has also explored the use of event structures to monitor interactions, particularly in the trust domain [8].

A large body of work in privacy-preserving data mining literature propose algorithms for k-anonymous data publishing problem. Most of these techniques can be classified as either generalization, suppression or a mixture of both techniques

[25], [32]. More recently, some work on location-privacy also incorporate similar measures for privacy [5], but none of them are directly applicable to our kind of application.

More recently, there has been some work on private searching on streaming data [24] which is of interest since we too model the pervasive space as a stream of events. The authors in [24] approach the problem in a similar manner, where the goal is to prevent the adversary from knowing whether a document matches the search criteria or not by obfuscating the pattern in which both matching and non-matching documents are handled. Though, the problem in our case is more complicated due to two main reasons, one because, we not only need to carry out the matching, but also update the state of the automaton. Second, while the authors in [24] do not address the problem of repeated access of the same document, we have to address this issue as we need to deal with a fixed set of automatons. This leads to additional inference channels not addressed in [24].

VII. Conclusion

In this paper, we addressed the privacy challenges that arise in human-oriented pervasive spaces when environmental data captured via sensing infrastructures (that may contain identifying information) is collected to drive pervasive functionalities. The core of such pervasive spaces is multi-modal event detection. We identified inference channels that arise in detecting multi-modal events that correspond to finite state automata of primitive events described over media streams. We developed an anonymizing methodology that explores the trade-offs between information disclosure and efficiency. While our paper represents a first such attempt at realizing pervasive functionality while preserving subjects privacy, our solution makes a few assumptions and simplifications. For instance, we assume that media event generation can be done error-free and without disclosing information to the server. While this is true for certain sensors (e.g., RFID events), in general, generating a media event may require sensors to communicate with the server; our prototype solution assumed anonymous sensor-server communication [4]. We focused on only the detection of media events scoping out the issue of action execution. The visibility of actions that execute as a result of the event detection might have privacy implications. We are currently expanding our work to incorporate multi-person events, address the privacy implications of "action execution" and extend our approach to deal with other notions of privacy besides anonymity.

References

[1] Insider attacks. http://csrc.nist.gov/publications/nistir/threats/subsection3_4_1.html, 1994.

[2] Responsphere. http://www.responsphere.org, 2007.

[3] J. G. M. V. A. Machanavajjhala, D. Kifer. L-diversity: Privacy beyond k-anonymity. In *ICDE 2006*.

[4] J. Al-Muhtadi, R. Campbell, A. Kapadia, M. D. Mickunas, and S. Yi. Routing through the mist: Privacy preserving communication in ubiquitous computing environments. In *ICDCS'02*, page 74. IEEE Computer Society, 2002.

[5] L. L. B. Gedik. Location privacy in mobile systems: A personalized anonymization model. In *ICDCS*, 2005.

[6] R. Campbell and et. al. Towards Security and Privacy for Pervasive Computing. In *ISSS 2002*, 2002.

[7] A. Demers, J. Gehrke, M. Hong, M. Riedewald, and W. White. Towards expressive publish/subscribe systems. In *ICDE2006*, 2006.

[8] C. English, S. Terzis, and P. Nixon. Towards self-protecting ubiquitous systems: Monitoring trust-based interactions. In *Personal and Ubiquitous Computing. 9(6)*, 2005.

[9] D. A. Fidaleo, H.-A. Nguyen, and M. Trivedi. The networked sensor tapestry (nest): a privacy enhanced software architecture for interactive analysis of data in video-sensor networks. In *VSSN '04: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 46–53, New York, NY, USA, 2004. ACM Press.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[11] S. Gatzui and K. Dittrich. Detecting composite events in active database systems using petri nets. In *4th RIDE-AIDS (2-9)*, 1994.

[12] B. Hore, H. Jafarpour, R. Jain, S. Ji, D. Massaguer, S. Mehrotra, N. Venkatasubramanian, and U. Westermann. Design and implementation of a middleware for sentient spaces. In *Proceedings of ISI'07*, 2007.

[13] B. Hore, J. Wickramasuriya, S. Mehrotra, and N. Venkatasubramanian. Privacy-preserving event detection for pervasive spaces. In *UCI-ICS technical report*, 2007.

[14] IBM. S3-R1: The IBM Smart Surveillance System – Release 1 (White Paper). http://www.research.ibm.com/peoplevision/IBMS3-R1Overview.pdf.

[15] A. Juels and R. Pappu. Squeling Euros: Privacy Protection in RFID-Enabled bank notes. In *Financial Cryptography*, volume LNCS 2742, pages 103–121. Springer-Verlag, 2003.

[16] D. Kundur, W. Luh, U. Okorafor, and T. Zourntos. Security and privacy for distributed multimedia sensor networks. *Proceedings of the IEEE*, 96(1):112–130, Jan. 2008.

[17] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE 2007*.

[18] A. Lubiw. Some np-complete problems similar to graph isomorphism. In *SIAM Journal of Computing*, Feb. 1981.

[19] M. Langheinrich. Privacy by Design: Principles of Privacy-Aware Ubiquitous Systems. Presented at ACM UbiComp 2001, Atlanta, GA 2001.

[20] MIT Project Oxygen. Pervasive Human-Centered Computing. http://oxygen.lcs.mit.edu/.

[21] R. Nevatia, J. Hobbs, and B. Bolles. An ontology for video event representation. In *CVPRW*, page 119, 2004.

[22] E. M. Newton, L. Sweeney, and B. Malin. Preserving privacy by de-identifying face images. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):232–243, 2005.

[23] V. Oleshchuk. Privacy Preserving Monitoring and Surveillance in Sensor Networks . In *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*, pages 485–492. Springer, Canada, 2007.

[24] R. Ostrovsky and W. E. S. III. Private searching on streaming data. In *CRYPTO, volume 3621 of Lecture Notes in Computer Science, pages 223–240. Springer, 2005*, 2005.

[25] P. Samarati and L. Sweeney Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression. Technical report, As technical report, SRI International, 1998.

[26] P. R. Pietzuch, B. Shand, and J. Bacon. A Framework for Event Composition in Distributed Systems. In *Proc. of the 4th Int. Conf. on Middleware (MW'03)*, Rio de Janeiro, Brazil, 2003.

[27] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Mobile Computing and Networking*, pages 32–43, 2000.

[28] M. Roman and R. H. Campbell. Providing middleware support for active space applications. In *Middleware 2003*, 2003.

[29] B. Schilit, A. LaMarca, G. Borriello, W. Griswold, D. McDonald, E. Lazowska, A. Balachandran, J. Hong, and V. Iverson. Challenge: ubiquitous location-aware computing and the "place lab" initiative. In *Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, pages 29–35. ACM New York, NY, USA, 2003.

[30] A. Senior, S. Pankanti, A. Hampapur, L. Brown, Y.-L. Tian, A. Ekin, J. Connell, C. F. Shu, and M. Lu. Enabling video privacy through computer vision. In *Security & Privacy Magazine, IEEE, Vol.3, Iss.3*, pages 50–57, 2005.

[31] D. Song, D. Wagner, and A. Perrig. Practical Techniques for Searches on Encrypted Data. In *2000 IEEE Symposium on Research in Security and Privacy*, 2000.

[32] L. Sweeney. k-anonymity: a model for protecting privacy. In *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, volume 10 (5), pages 557–570, 2002.

[33] D. E. D. Vinkemeier and S. Hougardy. A linear-time approximation algorithm for weighted matchings in graphs. *ACM Trans. Algorithms*, 1(1):107–122, 2005.

[34] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *Security in Pervasive Computing*, volume 2802 of *Lecture Notes in Computer Science*, pages 201–212, 2004.

[35] J. Wickramasuriya, M. Datt, S. Mehrotra, and N. Venkatasubramanian. Privacy protecting data collection in media spaces. In *ACM Multimedia Conference*, pages 48–55, 2004.

[36] BROADCOM secure applications processor - bcm5890 http://www.broadcom.com/products/Enterprise-Networking/Security-Processor-Solutions/BCM5890

[37] IBM PCI cryptographic coprocessor, 2004. http://www.ibm.com/security/cryptocards