Privacy-Preserving Event Detection in Pervasive Spaces

Bijit Hore, Jehan Wickramasuriya, Sharad Mehrotra,
Nalini Venkatasubramanian, Daniel Massaguer
School of Information & Computer Science, University of California Irvine

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

Emerging sensing, embedded computing, and networking technologies have created opportunities to blend computation with the physical world and its activities. The resulting pervasive environments offer numerous opportunities including customization (e.g., personalized advertising), automation (e.g., inventory tracking and control) and access control (e.g., biometric authentication, triggered surveillance). Multimodal event detection is an integral component of pervasive environments. Such systems capture and process raw streams of data (e.g. video, speech) and then convert them into semantically meaningful events. For instance, the entry of an unauthorized person (detected from a video stream) into a sensitive region may raise an alarm. Such events, once detected, may result in further actions that realize the functionality of the pervasive space.

Event-driven approaches have recently become a popular paradigm in which pervasive applications are implemented. The VERL system developed a language to specify and detect multimedia events [19]. The IBM $S3$ smart surveillance system [13] offers event-based retrieval in order to manage surveillance data. The Cayuga system and accompanying language, CESAR [6] proposed an event stream language and detection system that uses finite state automata to realize the detection of complex events for publish/subscribe applications [22]. An event-based approach offers a general framework using which a wide variety of pervasive applications can be built.

In this paper, our focus is on human-centric pervasive spaces. In such spaces, the collected environmental data may include personalizing information about individuals immersed in the space. This naturally leads to concerns of privacy. For instance, information about a person's location used to customize the space could potentially be misused as evidence of his/her presence/absence at a given location and time (whether or not it is in their best interest). Such privacy concerns arise when users do not fully trust the pervasive environment. Measures to establish trust, such as explicit policies to prevent leakage or sharing of personalizing information may alleviate, but do not eliminate such concerns. Indeed, numerous studies have identified such privileged users as the primary source of corporate data thefts.

If the pervasive space is untrusted, event detection systems must address a larger challenge; that of event detection from multi-modal streaming data without violating the privacy of individuals captured in the streams. Note that if the pervasive space is entirely untrusted, the goal of privacy preserving event detection will remain elusive. In our approach, we assume the presence of tamper-proof sensing devices capable of limited computation that can be programmed to generate a stream of events while hiding the raw signals from which such events are generated. Such an assumption is not unreasonable given advances in sensor technologies (e.g. research on low-cost cryptographic schemes [25], [14], [29]; smart surveillance systems such as IBM's $S3$ [13] already employ tamper-proof sensors. Privacy-preserving techniques, such as those that manipulate raw video stream data, (e.g face masking, removing/replacing identities) can be incorporated into the capture devices themselves [30]. These techniques are effective when one wants to detect simple events that can be evaluated based on the current event data itself. For instance, capturing the entry of unauthorized personnel into a protected space can be accomplished through credential-driven access control. However, for more complex events which requires one to store past event data, the problem is not so simple. Consider the case where one wants to detect repeated entries (say more than ten) of an employee into a certain room within a 8 hour window; here, a record of all previous entries in that window must be maintained. A significant amount of raw data from multiple sensors may be required for evaluation and handling this data securely becomes a critical issue that needs to be addressed.

Complex event detection on streams can be implemented by either storing the incoming data in the form of a log and evaluating predicates on it, or by taking an automaton-based approach as proposed in the Cayuga system [6]. Either way, the question is where and how to store the data or the automata in an untrusted environment? Sensors being the only trusted components in our model, would be the first candidates for secure storage of data. Instead, we argue that a centralized system is more viable for the following two reasons: (i) A complex event may evolve over an extended period of time

and might be distributed over spatial regions. For example, say we want to detect the event when a particular individual comes into the third floor of a building for the 4th time within a day. If the floor has multiple access points, the individual might enter the floor via different access points. Such a distributed event-detection involves multiple sensors accessing a common set of records (or automata) to correctly determine when the specified individual enters the floor for the 4th time. A centralized architecture is therefore more convenient. (ii) Also, a centralized database offers greater scalability than a distributed set of sensors with limited storage capacity. On the down side, the central server is neither a tamper-proof hardware, nor does is reside in a trusted environment. This makes the information stored on the server vulnerable to attacks. In this paper, we develop an automaton based approach for event detection, where the challenge lies in designing a secure scheme for evaluating automata on the sensor-generated stream of events. The security goal being that no confidential information regarding the nature of events should be available to entities in the untrusted regions. We outline the contribution of this paper below.

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 in 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 (Section V). We present some related work in Section VI and conclude in Section VII with some discussion and outline some open problems.

II. AN EVENT MODEL & SYSTEM ARCHITECTURE

We envision a pervasive environment as a physical space with an embedded sensing infrastructure which is used to monitor its state. The pervasive space is modelled as a set of users (U), a set of physical regions or space (P), and resources (R). Sensing in the pervasive space happens through a variety of mediums (e.g., video cameras, radio-frequency identification (RFID), motion detectors etc.). We refer to the raw sensor data gathered by the sensors as *media streams*. A media stream S is analyzed to extract *media events*.

Media Events: A *media event* is modelled as a timestamped 4-tuple, $e = (u, s, r, a) : t$ where u corresponds to a user (subject), s denotes a region, r denotes a resource, a specifies 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 in a surveillance setting represents that "Bob" entered the "loading dock" at "3:00pm". The resource here, could be anything (*) (could be of null value). With media events e, we will associate the following notation. We will refer to the user u as the user associated with event e. Furthermore, a particular instantiation of s, r, a in the event e will be referred to as *type of* the media event. For instance, in the above example, Bob is the user associated with the media event, and the event is of the type (loading dock, *, entry). The set of possible media events depend upon the pervasive application and are limited by what the sensing infrastructure can detect. 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. We also notice that the set of all media events that can be generated in this space is a finite, enumerable set. For instance, a media event belongs to 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-B and IV.

Primitive and Composite Events: A pervasive space is associated with a set of policies (or rules). A policy consists of a description of an event and the corresponding action that must occur if the event is detected. An event could either be *primitive* or *composite*. A *primitive event*, similar to the media event is modelled as 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, * 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. Consider, a media event (BOB, ROOM 113, BRIEFCASE, ENTRY):16:46 that represents the fact that "Bob" walked into room 113 carrying a briefcase at 4:46PM. Such a media event will match 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", "briefcase" and "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 [22], [6].

Composite Event Detection Composite events are translated

---

[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.

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.

Let us illustrate the automaton execution model via the use of an example. Consider a shared "smart" office space setting which is used by a number of research groups and staff of the office. The area is instrumented with various sensors (that monitor inhabitants, resource usage/consumption etc.). For example, workers wear RFID badges [30] to inform the building of their movements. We illustrate our event and execution models with the following application scenarios.

Scenario 1: (Surveillance) is done in the server room to detect if any member of STAFF enters the server room, and accesses the payroll database. Note that $\overline{u}$ refers to the instantiated value of u. We identify the following primitive events[2].

$$e_1 \equiv\, <u \in \text{STAFF}, \text{SERVER\_ROOM}, *, \text{ENTRY}>$$
$$e_2 \equiv\, <\overline{u}, \text{SERVER\_ROOM}, *, \text{EXIT}>$$
$$e_3 \equiv\, <\overline{u}, \text{SERVER\_ROOM}, \text{PAYROLLDB}, \text{ACCESS}>$$
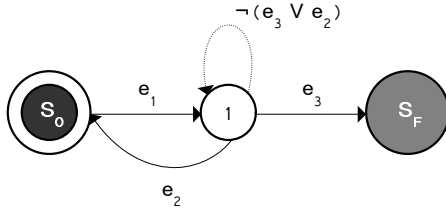

Fig. 1.   Automaton for scenario 1.

Scenario 2: (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. This is an example of a more distributed automaton where we need to look at two possible paths that can trigger the transition to the final state. The same notation as Scenario 1 applies here.

$$e_1 \equiv\, <u \in \text{DATABASE}, \text{FLOOR\_2}, *, \text{ENTRY}>$$
$$e_2 \equiv\, <u \in \text{DATABASE}, \text{FLOOR\_4}, *, \text{ENTRY}>$$
$$e_3 \equiv\, <\overline{u}, \text{FLOOR\_2}, *, \text{EXIT}>$$
$$e_4 \equiv\, <\overline{u}, \text{FLOOR\_4}, *, \text{EXIT}>$$
$$e_5 = e_6 \equiv\, <\overline{u}, \text{STORAGE\_ROOM}, \text{PROJECTOR}, \text{ACCESS}>$$

[2]* designates a wildcard and refers to the fact that there is no constraint on the resource or time in matching this primitive event
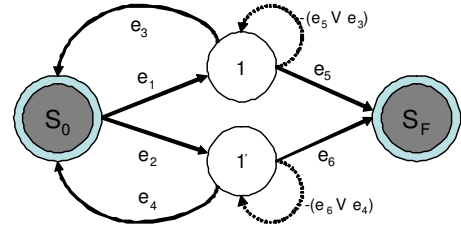

Fig. 2.   Automaton for scenario 2.

Now consider the following incoming media event; {Alice, server_room, laptop, entry}: 09:13. This media event instantiates the automaton in Fig. 1 by binding Alice to it (assume that Alice is indeed identified to be a member of the group STAFF). At this point, the automaton is well-specified and bound to Alice (i.e. $\overline{u}$ = Alice).

We make following observations about our automata:

- For every state of the automata, there is always a transition for all possible media events. Let F be an automata, s be any state of F, and let $t_1, t_2, \ldots, t_k$ be the set of transitions associated with state s of F with the corresponding primitive events p_1, p_2, $\ldots p_k$. For all media events e there exists a primitive event p_j, $1 \leq j \leq k$, such that p_j matches e. We need the above condition since our automata are utilized for pattern recognition in an event stream. 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 [6], 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 automata 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 automata F, we define the notion of $USER(F)$. $USER(F) \subseteq U$, is the set of individuals in the system 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, "Alice" would be in the $USER(F)$, for the automata depicted in Fig. 1. In contrast, "Bob" would not be in $USER(F)$ if "Bob" was not a member of the staff. We generalize the concept of $USER$ to the set of automata. Let F_S be an automaton set. $USER(F_S) = \bigcup_i USER(F_i)$, $F_i \in F_S$. For a given user u, we denote the set of automata for which u is in the $USER$ set as the automata associated
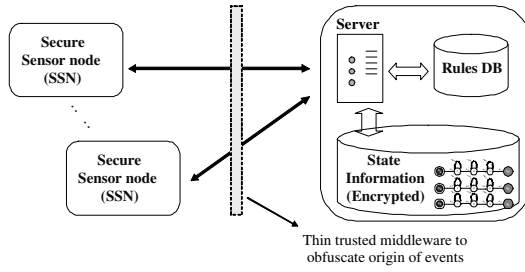
Fig. 3. System Architecture

with the user u. An automaton F is associated with a user u if $u \in USER(F)$.

A. System Architecture

Our event detection system consists of multiple components (depicted in Fig. 3). 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 [30], [13] (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 extender version of the paper [12]). 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.

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. (The server is responsible for generating messages for the "action executor" when a complex event of interest is determined, but we do not discuss that aspect in this paper).

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[3]. By default, each automaton in the state-table is in its start-state. After an automaton reaches the final state (and suitable action is taken) it is reset to its start state. Since the encryption scheme used is non-deterministic in nature, the start and final states are indistinguishable from any other state of the automaton. We also assume that there is no explicitly identifying information (like observable action-execution on reaching the final state for an automaton) available to an adversary that lets him discriminate one state from another.

Communication protocol $\mathfrak{P}$: Fig. 4 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. We make the assumption that, the automata can be "safely" tagged by the set of media events on which it can make a transition (from its current state), without revealing the media events. It retrieves a subset of rows from the state-table whose tags match the selection criteria (generally some equality predicate) specified by the SSN. It can carry out encrypted-matching if necessary (using secure keyword matching schemes like the one in [26]). 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 [26]).
- 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[4]) and sends them back to the server. If any automaton reaches its final state, the SSN notifies the server of event detection.

Note: A non-deterministic encryption is used which ensures that re-encryptions of the same plaintext are distinct from one another. 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.

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

---

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

[4] non-deterministic encryption ensures that multiple encryption of the same plaintext generate different ciphertexts.
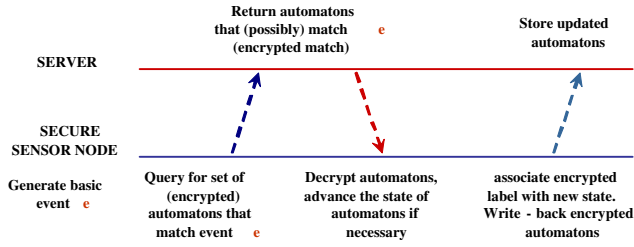
Fig. 4. A generic protocol to service events.

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 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 [21], [27] 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 (e.g., "disease = AIDS"). This led to more stringent criteria such as l-diversity [2], t-closeness [15] 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 wether 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.

A. Security & Adversary 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. There maybe one or more entities on the server-side (e.g., database administrators) who are regarded as adversaries. We assume a *passive adversarial* model, where the adversary is only interested in gleaning information about the events, and does not disrupt the functioning of the system in general. We argue that, the passive adversarial model is the most appropriate for our application because of the following reasons: (i) Probabilistically speaking, active adversaries are more likely to be caught if they change the state of the system that affects/disrupts its functionality, whereas passive adversaries are much more likely to go un-noticed, and therefore they are more probable. (ii) Most real-life incidences of privacy-breach are due to passive adversaries (insiders), who leak sensitive information without getting detected (right away). Though, this information maybe used in other contexts later to "actively" harm the owner, the adversarial model is a passive one. (iii) Majority of the literature in research also consider the adversary to be passive, and design solutions in the same which is considered as the most appropriate model. E.g., keyword-search over encrypted text data in remote email storage applications [26], Database-as-a-service model [11], most of the work in statistical databases, etc.

Additionally, 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.

B. 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 after 7:00 pm, 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. 4. 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. ◇

In general, an event e may be identified by determining one or more of the automata retrieved while servicing e. But, in order to identify an automaton, one might need to look at how it is being accessed over a period of time (i.e., over a long sequence of events). This is what we refer to as a *characteristic access pattern* of an automaton. Depending on the structure and transition edges in an automaton, its access patterns may have unique signatures ("row-access patterns") even when all the rows in the state-table are encrypted. Here is another example that illustrates what comprises an unique (observable) signature of an automaton.

Example: Let there be a set of 5 automata $\{A^*, A_1, A_2, A_3, A_4\}$ that make a transition on event e_1. Let there be another set of 5 automata $\{A^*, A_5, A_6, A_7, A_8\}$ that make a transition on e_2. Then A^* is the only automaton that can make a transition on both e_1 and $e2$. Also, assume that there are no other events (besides e_1 and e_2) that are common to any set of 5 automata. Now, if the adversary sees a row (in the row-encrypted state table) that is accessed along with 2 distinct sets of 4 automata (rows), he can be sure that this row corresponds to A^*. As a result, this characteristic access pattern of A^* has an unique signature and is therefore, identifying. ◇

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.

1) Access patterns & characteristic 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.◇
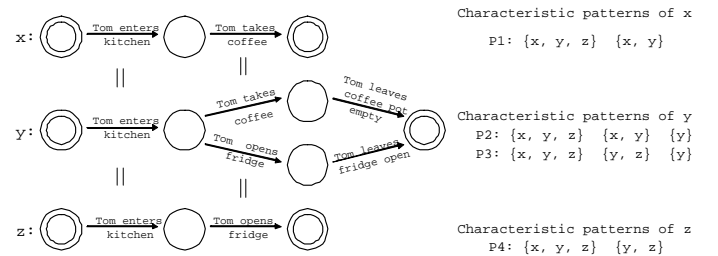

Fig. 5. Characteristic patterns.

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[5] 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.

Observable indistinguishability: Two automata are said to be *observably indistinguishable* or simply indistinguishable if

---

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

their characteristic sets are *identical*. We 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 $\forall$ automata $a \in \mathcal{A}$ 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 of automata 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 patterns of the encrypted automata. We will call such automorphism (if it exists), a *non-identifying automorphism*.

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. He can only identify an automaton by its characteristic access patterns. We now state the necessary and sufficient condition for achieving k-anonymity when the adversary can only see the characteristic patterns. We will 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) $E_0(i).USER \neq E_1(i).USER \neq \ldots \neq E_{k-1}(i).USER$, $\forall i = 1, \ldots, n$.

Proof outline: An adversary can only infer the identity of the individual involved in an event $e_i \in E_0$ by identifying the associated set of automata (which are accessed on e_i). Since the only observable features about the encrypted automata are their access patterns, if the set of characteristic patterns of an automaton are non-identifying, then it implies anonymity at the event-level as well, i.e., $e_i.USER$ cannot be determined. Condition (I) in the theorem is equivalent to the following fact: "For any row in the state-table, its observed characteristic pattern (embedded in $p(E_0, G_0)$) should not allow an adversary to

identify the corresponding automaton uniquely". Condition (II) specifies the "degree" of indistinguishability that is required. It ensures that the **k-anonymity** criteria (criteria 1) is satisfied: "The diversity of the set of automata up to which an encrypted automaton can possibly be identified is at least k".

From the adversary's viewpoint, the k sequences $(E_0, \ldots, E_{k-1})$ are indistinguishable. In other words, for each of these k sequences if there exists an assignment (map) of automata to rows such that for all rows in the table, the k automata that are mapped to it (in these k instances) are observably indistinguishable[6] from each other and correspond to distinct individuals, then the k-anonymity criteria is met. Therefore checking for the k-anonymity condition is same as determining if k "suitable" pattern automorphisms exist on the set of automata. $\diamond$

The problem of detecting whether a given set of automata satisfies the k-anonymity conditions is difficult. Even the case of 2-anonymity (i.e., when $k = 2$) is NP-Complete as shown in the following theorem. (Our conjecture is that the problem is hard for other values of k (> 2) as well.)

Theorem 2: (**Hardness of checking for k-anonymity**): The problem of recognizing whether a given set of automata (along with the corresponding set of events) is 2-anonymous is NP-Complete.

Proof outline: The NP-Complete problem of "**Deciding whether a graph G has a fixed-point free automorphism**" can be reduced to an instance of the problem of "Detecting a fixed-point free pattern automorphism on a set of automata". The reduction is presented in appendix B. $\diamond$

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 to ensure that the access patterns of the automata are k-anonymous. In the next section, we provide solution that meet these constraints.

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:

Solution 1: (**k-Individuals Partitioning**) Let $M = \lfloor N/k \rfloor$. Make M disjoint groups of individuals, where each group has at least k individuals. Assign all the media events appearing in any automaton corresponding to an individual within a group to a single cluster.

Note, that the above scheme assigns a static label to each automata 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 in a partition, thus leading to a homogenous access pattern for all automata within a partition, in effect 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

---

[6]There exists sufficiently many pattern automorphisms on the set of automata.

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_x^I, F_y^I) where both automata have at least one common transition (β edge) between states.

Example: Fig. 6 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".
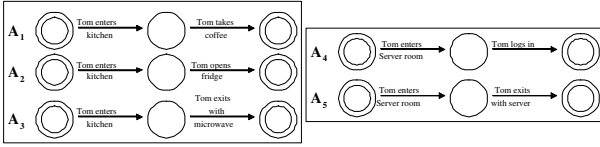

Fig. 6.   Example: Connected Groups

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 the outlined below.

Solution 2: (**k-connected-group Partitioning**) Partition the connected-groups of automata into clusters such that each cluster has diversity $\geq$ k (i.e., has automata representing at least k different individuals). Assign all the media events appearing in automata within a bin to a single cluster.

The above scheme guarantees k-anonymity as illustrated in the theorem below.

Theorem 3: (**Sufficient condition for k-anonymity**) Solution 2 is k-anonymous.

Proof: The scheme proposed above creates a partitioning of the set of automata such that automata belonging to different partitions (clusters) are never accessed simultaneously on any event, whereas all automata within the same partition are always accessed together. This implies that the rows in the state-table are indexed statically irrespective of the state they are in. As a result rows are accessed only at the cluster level (i.e., the SSN either retrieves all or none of the automata belonging to a cluster). Therefore, the characteristic patterns for all automata within a partition are the same making them observably indistinguishable from each other. E.g., If the events of automata, $A_1, \ldots, A_m$ (say, corresponding to rows $r_1, \ldots, r_m$ in DB) are grouped together, then each $A_i, i = 1, \ldots, m$ has exactly one characteristic pattern of any given length l ($l = 1, \ldots, \infty$), where each element in the sequence corresponds to simultaneous access of all the m rows, e.g., $S_l = \{\{r_1, \ldots, r_m\}, \{r_1, \ldots, r_m\}, \ldots l \text{ times }\}$. Each partition can be seen as a generator of one such class of patterns that is characteristic of every automaton in that partition. It is easy to see, that there is no way for an adversary to distinguish one automaton from another within a cluster using any observable access patterns. Such a scheme will meet the k-anonymity criteria of theorem 1. $\diamondsuit$

Thus, the above partitioning scheme may assign two connected groups of automata corresponding to the same individual to two distinct partitions[7]. 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 7.
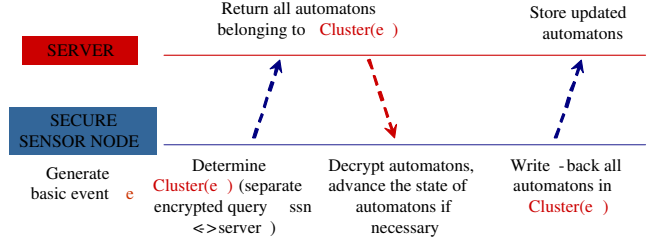

Fig. 7.   Secure protocol for servicing events.

Difficulty of achieving k-anonymity using more flexible schemes: The NP-Completeness result makes it computationally infeasible to design a more efficient partitioning scheme that achieves k-anonymity in the general case (where automaton structure may correspond to arbitrary directed graphs with cycles). For instance, consider the following scheme:

"*Partition the set of all media events into bins of size at least k, such that each bin has events representing k different individuals (each bin is k-diverse).*

Unlike solution 2, where all media events appearing in an automaton are forced to be in the same partition, the above approach is more flexible and is a superset of the set of partitioning schemes resulting from solution 2. The question is whether such a k-diverse event-clustering scheme achieves k-anonymity? The answer is no! Such a partitioning scheme transforms the set of automata by effectively adding many self-transitions to the states (nodes) and does not reduce the complexity of the "k-anonymity verification" problem as such. In appendix A we use an example to illustrate how such a k-diverse partitioning scheme fails to achieve k-anonymity. The reason being, the characteristic set of the modified automata may still be unique and therefore allow an adversary to infer its true identity by observing row-access patterns in our model. (For security analysis, we need to assume that the event partitioning scheme is public knowledge and therefore known to the adversary). It might be possible to come up with a more efficient solution in special cases where the composite events lead to simple automaton structures, but we do not pursue those issues here.

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

---

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

average *costs* (i.e., number of false-positive retrievals) than others. The goal is to partition connected groups of automata in order 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 cluster (bin) are to be accessed together, the associated *cost* of a solution scheme $\mathfrak{S}$ is given by the following expression.

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

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 [8] which is a known NP-hard problem, can be reduced to an instance of our balls-and-bins optimization problem. The reduction is shown in appendix section C. 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 from the same minimum sum of squares partitioning problem.

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 2nd 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 input the number of distinct individuals (colors) in the system which N 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 next as 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 [28]) can be utilized for computing the **most profitable** matching in the graph. Ball-exchanges[8] between bins can also be modeled similarly. Finally, since the ball transfers are only carried out for positive cost-reductions, the iterative algorithm always terminates. Finally, it can be seen that the algorithm terminates by noting that the last *while* loop in Algorithm 1 will definitely terminate since only positive cost-reductions are allowed.

V. IMPLEMENTATION & EVALUATION

We built a prototype of the system described in this paper on top of the SATware-Responsphere framework [10], [1]. SATware is a middleware framework for programming and testing

---

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

Algorithm 1 Least cost k-diverse partition.

1: Input:Color$[1, \ldots, |B|]$,Weight$[1, \ldots, |B|]$,Price$[1, \ldots, |B|]$, k, NUM_COLORS;
2: Output: Cost, Bin$[1, \ldots, |B|]$;
3: **if** k > NUM_COLORS **then**
4:      Print: "No solution possible for given k";
5:      **Return** ϕ;
6: **end if**
7: /* Generate an initial feasible solution */
8: $X \leftarrow$ set of all balls; $i \leftarrow 0$;
9: **while** k or more distinct colored balls in X **do**
10:     Initialize new bin B_i;
11:     Randomly put k distinct colored balls from X to B_i;
12:     $i \leftarrow i + 1$; $X \leftarrow X \setminus B_i$;
13: **end while**
14: Let $M \leftarrow i$; /* num bins initialized */
15: **if** $M = 1$ **then**
16:     **for** $i = 1$ to $|B|$ **do**
17:         $Bins[i] = 1$;
18:     **end for**
19:     $cost \leftarrow (\sum_{i=1}^{|B|} Weight[i]) \times (\sum_{i=1}^{|B|} Price[i])$;
20:     Return ($cost$,$Bins[1 \ldots |B|]$);
21: **end if**
22: **if** $\exists$ balls not assigned to any bin **then**
23:     Assign each such ball to a random bin B_i, $i \in [1, M]$;
24: **end if**
25: /* Iteratively carry out cost-reducing transfers & exchanges */
26: **while** 1 **do**
27:     $C \leftarrow \phi$; /* set of candidate transfers */
28:     **for all** i, j where $1 \leq i, j \leq M$ and $i \neq j$ **do**
29:         $C \leftarrow C \cup$ **best feasible ball transfer** from B_i to B_j with a positive cost reduction;
30:     **end for**
31:     **if** $|C| \geq 1$ **then**
32:         $P \leftarrow$ most **profitable** set of compatible transfers in C;
33:         Execute all transfers in P;
34:         Reflect new bin assignments in $Bins[1 \ldots |B|]$;
35:         Compute the new $cost$ of partitions;
36:     **end if**
37:     $C' \leftarrow \phi$; /* set of candidate exchanges */
38:     **for all** i, j where $1 \leq i, j \leq M$ and $i \neq j$ **do**
39:         $C' \leftarrow C' \cup$ **best feasible ball exchange** between B_i and B_j with a positive cost reduction;
40:     **end for**
41:     **if** $|C'| \geq 1$ **then**
42:         $P' \leftarrow$ most **profitable** set of exchanges in C';
43:         Execute all exchanges in P';
44:         Reflect new bin assignments in $Bins[1 \ldots |B|]$;
45:         Compute the new $cost$ of partitions;
46:     **end if**
47:     **if** $|C| == 0$ AND $|C'| == 0$ **then**
48:         /* No candidate transfers or exchanges */
49:         **Return** ($cost$, Bins$[1, \ldots, |B|]$);
50:     **end if**
51: **end while**

Fig. 8. The user's identity is concealed when he has his 1st (a) and 2nd cup of coffee (b), but revealed when he has his 3rd cup (c).

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, 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[9] in all and defined 15 rules over these groups. The instrumented part of the floor consisted of 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 are affected by any event that takes place within the kitchen and like-wise for the other two sets. (As a result, there could be at most three connected groups corresponding to each individual.). The 15 rules that we used for our experiments are listed below.

Kitchen Rules

1) An individual enters the kitchen and takes a cup of coffee from the coffee machine.
2) An individual enters the kitchen and moves the location of the coffee machine.
3) An individual enters the kitchen and exits with the coffee machine.
4) An individual enters the kitchen and exits with the microwave.
5) An individual enters the kitchen, opens the refrigerator and exits without closing the refrigerator door.

Supply-room Rules

1) An individual enters the supply-room and takes print-outs from the printer.
2) An individual enters the supply-room and exits with the scanner.
3) An individual enters the supply-room and exits with the projector.
4) An individual enters the supply-room and moves the location of the copier.
5) An individual enters the supply-room and moves the location of the printer.

Server-room Rules

1) An individual enters the server-room and moves the IBM server.
2) An individual enters the server-room and moves the DELL server.

---

[9]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.

3) An individual enters the server-room and moves the SUN server.
4) An individual enters the server-room and logs in using the DELL server.
5) An individual enters the server-room and moves the IBM server.

The corresponding automaton-structures are shown in figures 9, 10 and 11. We assigned rules to individuals by associating each of the 3 rule-sets to one or more of the 4 groups of individuals. That is, if KITCHEN rules apply to STUDENT and STAFF groups, the system will instantiate 5 automata for each individual in either one of these groups corresponding to the 5 KITCHEN rules. Additionally, we introduced some outliers by randomly assigning (de-assigning) some rules to a small fraction of individuals. For example, say for 5% of the students, 3 out of 5 of the kitchen rules might not be applicable, whereas they apply to the remaining 95%. The idea is that such outliers may exist in real scenarios.
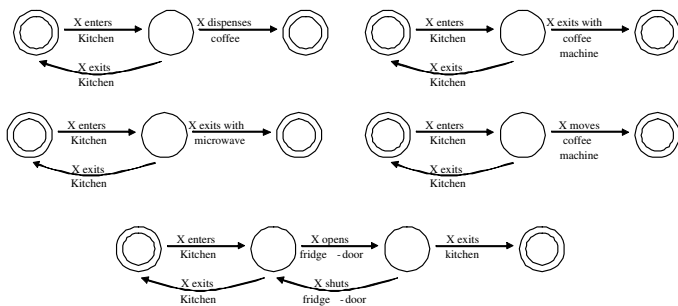

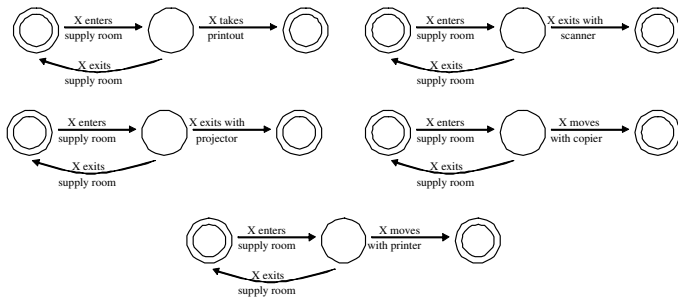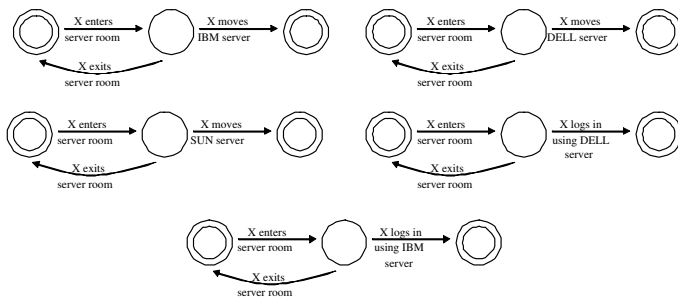Fig. 9.   Kitchen rules

Fig. 10.   Supply-room rules

Fig. 11.   Server-room rules

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. We describe some of these details later in this Section. Figure 8 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 clustering algorithm: To further study the performance and security/scalability tradeoffs of the clustering 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 clustering 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 12 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.
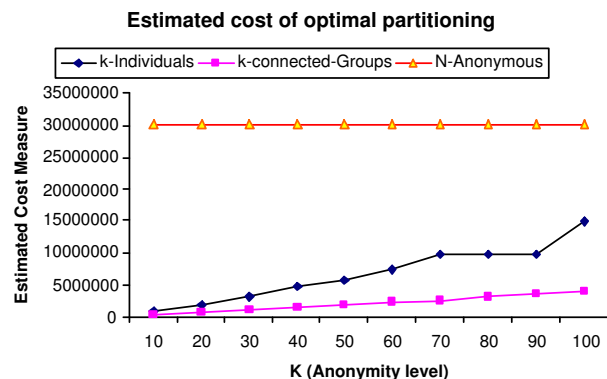

Fig. 12.   Cost estimates for the constrained ("Together") and unconstrained ("Separate") partitioning schemes

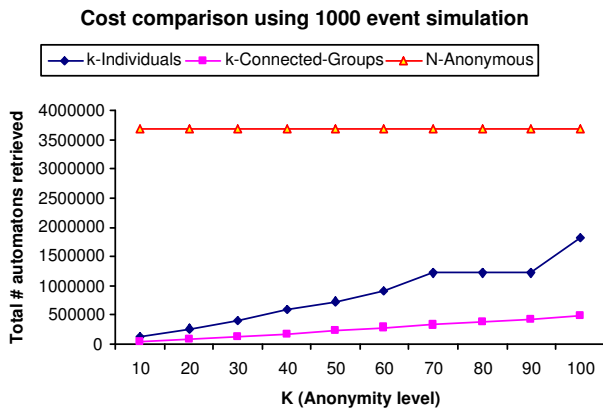For the second plot, we generated a sequence of 1000 media

Fig. 13. Total # automata retrieved in 1000 events

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. 14 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.
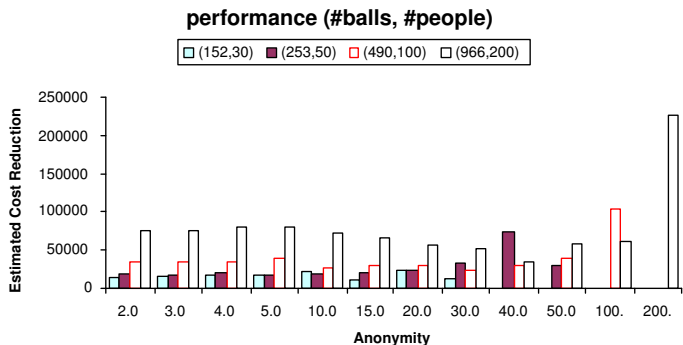
events as described above. Figure 13 shows load characteristics of the two clustering schemes on this test sequence. The y-axis is the total number of automata retrieved over the 1000 events which represents the transmission-load due to the "SSN-server communication" in a real setting.

The two important conclusions that can be drawn from observing the two plots are that: (a) The simulated transmission overhead in Figure 13 varies (grows with k) in almost an identical manner to what is predicted by the partitioning algorithm (cost estimate in Figure 12) 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 clustering 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 modeled 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

Fig. 14. Analytical estimate of performance improvement

B. Implementation Issues

Detection of event clusters at runtime: On generating a media event e the SSN needs to determine the cluster to which e is mapped. We implement this step using a secure encrypted keyword search scheme similar to that of [26] as follows: In a one-time pre-processing phase, the following two-column table T_{EC}(*media-event, cluster-id*) is generated (using *oblivious* interactions with the server) by one of the SSNs and stored on the server. T_{EC} is used only as a lookup table (read-only) subsequently. The first column of T_{EC} stores $E_{k_{SSN}}("e_i^j")$ which is the encrypted representation of the event e_i^j. The second column stores the "CLUSTER-ID" of the cluster to which e_i^j is mapped. Here $E_{k_{SSN}}$ denotes an encryption scheme using the secret key k_{SSN} common to all SSNs. Now, on generation of an event e_x^y, the SSN issues an encrypted query to the server for $E_{k_{SSN}}("e_x^y")$ and the match is carried out by the server on entries of Column 1 of T_{EC}. If a match is detected, the corresponding cluster of rows from the state-table (i.e., containing the encrypted automata belonging to the same cluster as e_x^y) are returned to the SSN. Such a retrieval model is secure and gives out no information about

the media event that was generated beyond the identity of the cluster to which it belongs.

In the oblivious pre-processing phase, the SSN simply writes the media-events into the table T_{EC} in a random sequence so as to ensure that the row-id and events are not correlated in any manner whatsoever beyond what can be gleaned from looking at the cluster-id of a row.

Concurrent access: It is quite possible that multiple sensors request the same group of connected automata at the same time (since automata corresponding to different individuals might be grouped together). Server will be required to employ a write-lock for this purpose. Some preliminary simulations indicate that the performance degradation is graceful with increasing load. Also, since the event generation in the pervasive space is driven by humans, natural latency of human activity does not pose a great challenge to performance.

Key-management in SSNs: The issues regarding key-management of SSNs is an important aspect. Recall, that we need a shared secret key (between all SSNs) that is used for and encryption/decryption of the automaton objects. We summarize our key-generation/update protocol below:

- Each SSN has a public-private key pair (pub_i, prv_i) and the list of public keys of each sensor is stored on the server.
- Periodically (could be daily) one SSN elects itself as the leader[10], say SSN_i and generates a secret key s_k for encrypting the automata.
- Leader communicates with the server to get public keys of all the SSNs. It then encrypts s_k with each pub_j, $j \neq i$ and communicates s_k to the respective SSN via the server. SSN_j can then decrypt and retrieve s_k using prv_j.
- The leader also executes another communication protocol with the server to re-encrypt all the current rows (automata) using the newly generated secret key s_k. In this step, the leader can optionally also carry out a random permutation of the row contents in the state table. (There are many possible algorithms for this, for instance, SSN_i may read two rows at a time from the state table and swap them while write back (after re-encryption) with probability 0.5.

VI. Related Work

There is a large body of work in the area of systems for pervasive computing environments [18]. GAIA [24] is an example of a system designed inherently to enable pervasive computing. Other systems such as CRICKET [23], 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 [5]. Langheinrich [17] 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 [3] combines hop-to-hop routing based on handles with limited public-key cryptography to preserve privacy from

---

[10]Conflicts can be detected using a locking protocol.

eavesdroppers and traffic analyzers. Our previous work [30] 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) [19] comes very close to the type of event detection we envision for pervasive spaces. VERL 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. Other types of event-based systems [9], [19] 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 [7].

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 [21], [27]. More recently, some work on location-privacy also incorporate similar measures for privacy [4], 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 [20] which is of interest since we too model the pervasive space as a stream of events. The authors in [20] 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 [20] 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 [20].

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 automatons 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 which we list next.

Media Event generation: We assumed that sensors (SSNs) can generate media events without disclosing information to the server. While this assumption is valid for certain sensor (e.g.,

events detected by RFID readers), in general, generating a media event may require sensors to communicate with the server. Current solutions does not ensure non-disclosure in such a case. We further note that a slightly different model for event generation is possible, in which the state of the automatons is instead stored on the distributed sensors directly (instead of being stored on the untrusted server). Such an approach is also interesting, and we believe that the solutions we have developed for a server-oriented protocol could provide insights into designing a distributed solution as well.

Anonymous Sensor-Server Communication: Identity of the sensor sending information to the server could lead to disclosure. In general, the server could infer additional information about the type of the incoming event (e.g. location, type of action, etc.) from the identity of the sensor sending the request. We have assumed that some anonymous sensor-server communication [3] is in place, which in reality still needs to be integrated into the current solution. This will potentially have an overhead on the performance. Note that this type of disclosure affects both the naive as well as the cluster-based algorithm.

Generalization of Media Events: We assumed state transitions in a specific automaton are instantiated by the events associated with a single individual. In general, we may also be interested in detecting complex events involving multiple individuals. For instance, we may wish to detect the composite event where any 4 people belonging to 4 different groups come together in a particular room. There are a number of different ways (combination of different individuals) in which this might occur. A secure implementation would need to have an automaton for each possible combination of 4 people belonging to the 4 groups. Additionally, one must consider which individual should be associated with each automaton. Similar issues arise if one has to detect a combination of multiple resources in a single media event (e.g. multiple items would be required to be tracked in Scenario 2, found in Section II), in which case the "detection of event-clusters" using encrypted matching becomes a challenge. Another interesting issue is when an actual single basic event can satisfy multiple primitive event specifications. In such a case, multiple events need to be generated by the SSN which can then form an alternative inference channel if the adversary can detect that these events are correlated in time. As one can see, the complexity of the event specification and detection increases as we start defining more varied rules. Extending our model to support such events is a key direction of our future work.

Rule Execution: 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.

Model of Privacy: We only considered anonymity as the definition of privacy in this paper. Extensions to our approach that address the problems of other notions of privacy are also very interesting.

Our current research is exploring approaches to relax the above assumptions which will significantly enhance the generality of our solution.

REFERENCES

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

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

[3] 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.

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

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

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

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

[8] 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.

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

[10] 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.

[11] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *International Conference on Very Large Databases (VLDB)*, 2004.

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

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

[14] 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.

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

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

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

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

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

[20] 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.

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

[22] 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.

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

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

APPENDIX

A. Example: k-diverse event-clustering

Example: Let there be three rules R_1, R_2 and R_3 and three individuals I_1, I_2 and I_3 such that rule R_i applies only to individual I_i. In our model, there will be one automaton object that is instantiated for each rule corresponding to the individual the rule applies to. Let these be denoted F_1^1, F_2^2 and F_3^3 respectively. These automaton objects are stored in 3 separate rows of the state-table (MAXSIZE = 3). Figure 15 shows these three automatons. Now consider the pattern-template $p = \{\{x\}, \{x\}, \{x\}\}$. It has 3 possible instantiations: $x \leftarrow F_1^1$ or $x \leftarrow F_2^2$ or $x \leftarrow F_3^3$. For each of these 3 instances, there is a sequence of events involving I_1, I_2 and I_3 respectively such that the template p can be realized. That is, there are sequences of media events e.g., $\{e_1^1, e_2^1, e_3^1\}$ which results in a single row being accessed three times in succession. But the template $\{\{x,y\}, \{x,y\}, \{x,y\}\}$ is not realizable since there exists no event which results in retrieval of two automatons (rows) simultaneously from this set.
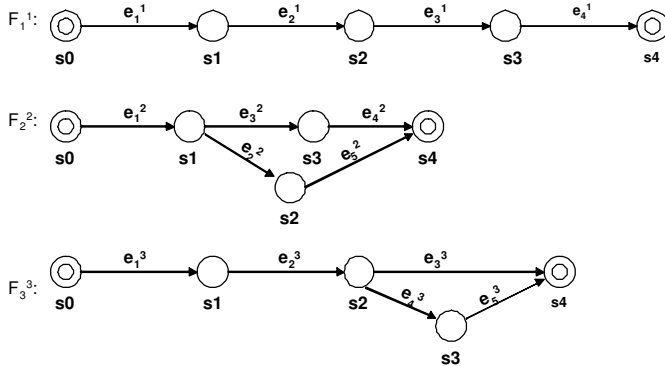

Fig. 15.   Example - Automatons and event sequences.

Let the following 5 event types be defined for media events, $E_{types} = \{e_1, e_2, e_3, e_4, e_5\}$. Also, assume 3 individuals are associated with the space and represented as: $\mathfrak{I} = \{I_1, I_2, I_3\}$. Then, the complete set of event instances E_{media} consist of the following 15 events $E_{media} = \{e_1^1, e_2^1, \ldots, e_1^2, \ldots, e_5^3\}$, where the superscript denotes the individual and the subscript denotes the event type. Let there be 3 distinct rules, R_1, R_2 and R_3 such that R_i applies only to I_i, i.e., there is only one instantiation of each rule. The three corresponding automatons are denoted by the set $F = \{F_1, F_2, F_3\}$ and shown in Fig. 15 (self-loops are not shown).

Now, consider the following clustering of the events into four 3-diverse clusters (i.e., all 3 individuals are represented in each of the clusters): $C_1 = \{e_1^1, e_5^2, e_4^3, e_5^3\}$, $C_2 = \{e_2^1, e_1^2, e_2^2, e_3^3\}$, $C_3 = \{e_3^1, e_3^2, e_1^3\}$ and $C_4 = \{e_4^1, e_4^2, e_2^3\}$. Fig. 16 shows the modified tags for the automatons. (Note that we do not need to consider events that do not appear in any rule, e.g., e_5^1 in the above case).
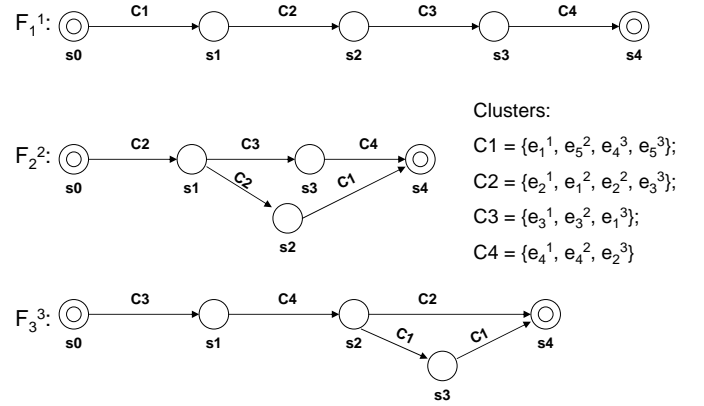

Fig. 16.   Example - Automatons indexed by event clusters.

Let the following sequence of two events be generated in the pervasive space: $S = \{(e_1^2 : t_1), (e_1^3 : t_2)\}$. At t_1 when e_1^2 is generated, the SSN will map the event to its corresponding cluster-id, which is C_2(later we will explain how this is done securely without revealing either the event or the cluster-id to the server). Subsequently, all rows currently indexed by C_2 are to be retrieved, decrypted, updated, re-encrypted and written back into the state table. At time t_1 just a single row is accessed, that corresponding to F_2^2 (which is in state $s0$). The state of the automaton is updated to $s1$, the row is indexed by two encrypted tags $E_k("C_2")$ and $E_k("C_3")$ denoting C_2 and C_3 respectively (for now assume that a searchable encryption scheme allows this without revealing the number of distinct tags, i.e., 2 in this case) and then the row is written back. At this point, an adversary is not able to determine which of the three automatons might have been initiated, hence 3-anonymity is ensured for all individuals. Now, at time t_2 when the second event e_1^3 is generated, all entries indexed by $Cluster(e_1^3) = C_3$ are retrieved. This set consists of two rows: one that was retrieved in the first step (corresponding to automaton F_2^2 currently in state $s1$) and the row corresponding to automaton F_3^3, currently in its initial state $s0$. In the write back phase, both the rows are written back after suitable updates and re-encryption is done. However, after the execution of the protocol at t_2, the anonymity of the individuals associated with the events at t_1 and t_2 are lowered from 3 to 2 by making the following

deduction (by working backwards): At t_0, all automatons would have been in their initial states, indexed by distinct cluster-ids (i.e., encrypted labels denoting C_1, C_2 and C_3 corresponding to the clusters of the transition edges out of the initial states of the 3 automatons) and any event would at most result in retrieving one row. Combining this information with the observation at t_2 where two automatons are retrieved, the adversary can deduce that the event at t_1 would have resulted in the state change of the retrieved automaton. He is now able to further deduce that at t_1, the automaton that was retrieved could not have been F_3^3, since then it would have been in state $s1$ before t_2 and indexed by C_4, which would never result in retrieval of two rows simultaneously at $t2$. By this analysis, the adversary can deduce that the event at t_1 could have only belonged to I_1 or I_2 (and not I_3) and the event at t_2 could have been that of I_2 or I_3 respectively (and not of I_1). This brings down the anonymity to unsafe level of 2. (Note that the inference is possible by simply observing the pattern of row accesses from the state table. The leakage happens in spite of encrypting the contents and index tags of rows in the table.) $\diamond$

B. Proof of NP-Completeness of k-anonymity detection

Here we present a proof of theorem 2 from section III. Actually, we show that for a candidate solution scheme $\mathfrak{S} = (\mathfrak{D}, \mathfrak{P}, \mathfrak{C})$ where $\mathfrak{C}$ denotes an event clustering scheme (section IV), the problem of checking whether it achieves k-anonymity is NP-Complete. We present a simple reduction of any instance of the problem of checking the existence of a fixed-point free automorphism in a graph G to an instance of our problem. The former is known to be a NP-Complete problem [16].

Reduction: Given a graph G = (V,E), generate an instance of our problem as follows:

- For each vertex $u \in V$, let there be an individual I_u associated with the pervasive space.
- For each $u \in V$, let generate an automaton A_u with two states s_u and f_u (denoting start and final states).
- For every edge $(u, v) \in E$, add two transition edges between start and end states of automaton A_u and A_v. Label these two transitions $C_{(u,v)}$.

The above construction gives rise to an instance of our problem containing a set of 2-state automatons. Since each automaton is assigned to an unique individual, common transition edges between two automatons can be interpreted as formation of 2-event clusters. For example, a transition edge labelled $C_{(u,v)}$ can be assumed to correspond to two events e_v^u and e_u^v in the automatons corresponding to individuals I_u and I_v respectively. Therefore, the reduction implicitly determines a set of appropriate events as well as an event-clustering scheme. Also, note that by the nature of the construction, each media event will lead to retrieval of exactly two automatons from this group. The following observation is critical.

Observation 1: The characteristic set of an automaton can be represented simply by the set of all its length-1 patterns. All longer patterns are simply concatenation of these length-1 patterns.

Since the set of edges in G completely specifies the set of all observed automaton access patterns, it is easy to see that the two problem instances are equivalent.
($\rightarrow$) Recall, under the definition of pattern automorphism, an automaton and its image (under this map) have the same set of observed patterns. But, since the set of length-1 patterns represent exactly the set of edges in G, existence of a fixed-point free automorphism in G will automatically imply the existence of a fixed-point free pattern-isomorphic (automorphic) map in the corresponding set of automatons (with respect to the event-clustering scheme implied by the reduction).
($\leftarrow$) The two problems being completely equivalent, the reverse implication also holds. $\diamond$

Fig. 17 shows an example of the reduction. The graph on left generates the set of 5 automatons on the right. The transition edges in the automatons are labelled by the cluster-ids of the event-clusters that are implicitly formed by this reduction. The 2 events comprising each cluster are shown on the right. Recall that the superscripts in each event e_i^j denotes the individual i to whom this event belongs. The subscript is simply used to denote different events.
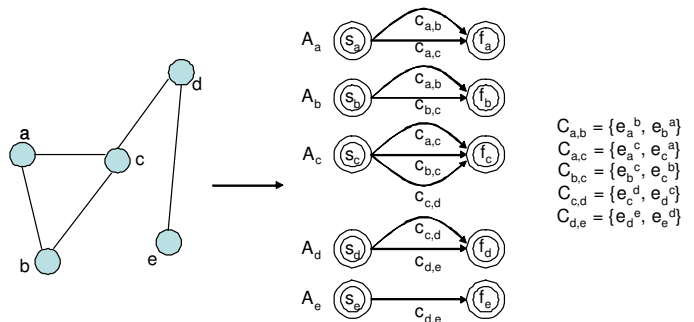

Fig. 17. Example: NP-Hardness reduction

C. NP-Hardness of minimum-cost partitioning of balls into bins

We show that the minimum cost ball partitioning problem of section IV is NP-hard by reduction from the "minimum sum-of-squares" problem [8]. The minimum sum-of-squares problem is the following.

Definition 7: (Minimum Sum-of-Squares) Given a finite set A, size $s(a) \in Z^+$ for each $a \in A$, and an integer $K \geq 2$, determine a partition of A into K disjoint sets $A_1, \ldots, A_K$, such that the following cost measure is minimized

$$\sum_{i=1}^{i=K} (\sum_{a \in A_i} s(a))^2$$

Now, any instance of the above mentioned problem can be reduced to an instance of our problem as follows: For each element $a \in A$, let there be a "RED" ball b_a with $weight(b_a) = price(b_a) = size(a)$. Also, introduce K "BLUE" balls, each with price = weight = 1. Then, an optimum partitioning of this set of balls that achieves 2-anonymity will also give an optimum solution to the corresponding "minimum sum-of-squares" problem.

The BLUE balls impose the required restriction on the number of bins. The monotonicity of the squares function along with the uniformity of the BLUE balls effectively cancel out the effect of their presence in each bin. ◇