# Privacy-Preserving Event Detection in Pervasive Spaces

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

***Abstract*— Pervasive applications often require gathering information about individuals that may be considered sensitive. Often, one is forced to make a difficult choice: either to risk loss of privacy or to forgo the benefits that pervasive technology offers. Our conjecture is that it is possible to design and deploy applications in a pervasive environment that do not come at the expense of individuals' privacy. In this paper, we consider 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 a systems architecture for deploying a surveillance application in such an environment. The privacy challenge corresponds to that of detecting only events that break certain rules without disclosing any identifying information unless necessary. We characterize the nature of various inference channels that arise in designing such a system and determine appropriate security constraints that need to be met. We model privacy-preserving event detection as an optimization problem that attempts to balance disclosure and performance in such a system, and design efficient communication protocols for our proposed architecture.**

## 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 [15]. The IBM $S3$ smart surveillance system [10] offers event-based retrieval in order to manage surveillance data. The Cayuga system and accompanying language, CESAR [3] proposed an event stream language and detection system that uses finite state automata to realize the detection of complex events for publish/subscribe applications [17]. 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 [21], [11], [25]); smart surveillance systems such as IBM's $S3$ [10] 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 [26]. 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 [3]. Either way, the question is where and how to store the data or the automatons 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 $4^{th}$ 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 automatons) to correctly determine when the specified individual enters the floor for the $4^{th}$ 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 automatons 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.

**Our contributions:** We model the environmental data as pervasive *multimedia streams*, which are processed into a stream of *media events*. One or more sequence of media events comprise a *composite event* which in turn are used to implement the functionality of the pervasive space. *Rules* of the space are defined as *(composite-event, action)* pairs. When a certain composite event is detected, the corresponding action is taken. In the current work, we concentrate on the event detection part of a rule. We propose a data model and a system architecture to detect such events in the pervasive space. We also characterize the nature of privacy and "channels of disclosure" in such an environment. Subsequently we design a secure communication protocol between trusted and untrusted components that enables (composite) event detection. 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 modelled 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.

**Paper Layout** The remainder of this paper is organized as follows. Section II presents our model of the pervasive space, and a description of our event model. Section III discusses our proposed system architecture and introduces the issue of privacy. Section IV presents a formal analysis of privacy-risks in designing of such a system. In Section V, we develop a protocol that attempts to balance disclosure and performance in building a privacy-preserving event detector. We also provide a discussion on the various assumptions and constraints used in developing this solution, and ways to relax them. Section VI provides a preliminary experimental evaluation of our protocol, and discusses performance issues. We present some related work in Section VII and conclude in Section VIII with some discussion and outline some open problems.

## II. An Event Model for Media Spaces

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 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 $t_\theta$, 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. $\theta$ 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 [17], [3].

**Composite Event Detection** Our goal is to detect composite events over the incoming stream of media events. Composite events are translated into their corresponding finite state automata (FSA) for this purpose. Composite events can be detected by maintaining a FSA corresponding to the regular expression comprising the event. Based on the events seen so far in the stream, the automaton maintains all "partially matched sequences". We utilize a traditional non-deterministic FSA model with some slight extensions. 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 automata executes as follows. An automata, when initiated is in the initial state $s_0$. At any state $s_i$, an automata 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) the automata 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 [26] 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].

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

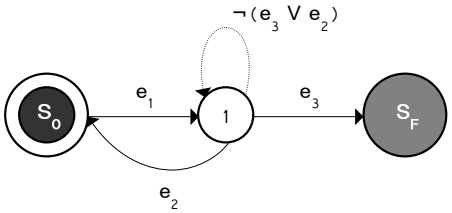


Fig. 1. Automaton for scenario 1.

*Scenario 2:* (Inventory Tracking) is done on two floors of a building to determine if any member of the *crypto* research group enters either of these floors and then leaves with a projector. 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.

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

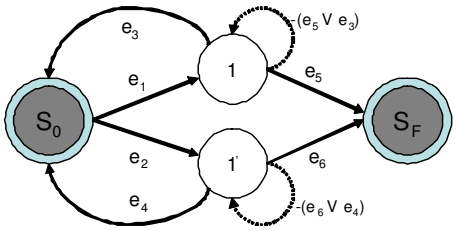


Fig. 2. Automaton for scenario 2.

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 automatons are utilized for pattern

[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.

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

recognition in an event stream. We differentiate between edges of the automata that are used to drive the automata to its successor state ($\alpha$ edges) from the edges that are self-loops used to filter out media events unrelated to the progression of the automata ($\beta$ edges). A similar concept of filter edges was used in [3], which consume events that do not drive a particular automaton to advancing its current state. Note that while implementing automatons of this kind, we can effectively ignore $\beta$ edges as they do not alter the state of the automatons. We will henceforth ignore $\beta$ edges of automatons.

- The automatons 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 $\alpha$ 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 automatons. 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 automatons for which $u$ is in the $USER$ set as the automatons associated with the user $u$. An automaton $F$ is associated with a user $u$ if $u \in USER(F)$.

## III. Architecture & Privacy Goals

Our event detection system consists of multiple components (depicted in Fig. 3). The two primary components of the event detection system are (1) the set of secure sensor nodes (SSN), and (2) an untrusted server with large storage and computational capacity that stores state information about the pervasive space:

**SSNs:** The secure sensor nodes are also the media event generators that convert the media stream into sequence of corresponding media events. The SSNs are trusted and assume to be tamper proof. The sensor nodes 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 [26], [10] (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 will discuss its implications as well as methods to resolve the resulting privacy challenges later (in Section VIII).

**Server/State Manager:** This denotes our central server, which 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 also 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).

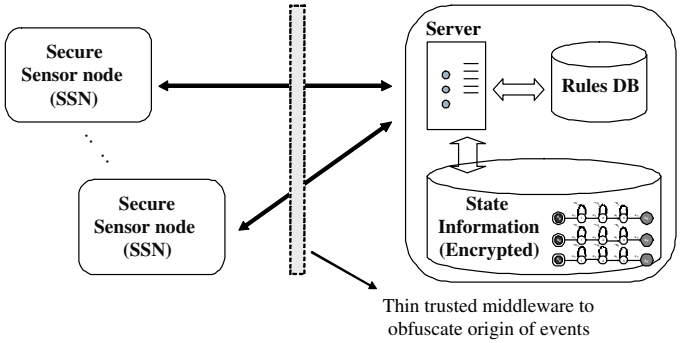


Fig. 3. System Architecture

### A. Adversary Model

An adversary in our model is one who wants to know what events are occurring in the environment (especially the identity of the individual involved in every media event that is generated by the SSNs). The key challenge we address is this paper is that the server-side environment is untrusted. 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 [22], Database-as-a-service model [8], most of the work in statistical databases, etc.

Additionally, we will assume the following background knowledge to be available to an adversary.

**Background knowledge**: The adversary knows the set of events that can be generated in the pervasive space and the rules that apply to individuals in this space. In other words,

he knows (i) the set of media events that can be generated by the sensors, (ii) the set of individuals that interact with the space, (iii) the rules-to-individuals mapping (i.e., which rules apply to which individuals) and (iv) the details of the automatons that implement these rules.

### B. Privacy Goals

Here, by privacy we will mean *anonymity*. Anonymization is a special case of *disclosure protection*. Whereas the goal of the former technique is to avoid exposing the identity of individuals, the latter refers to a more generic notion of exposure of information where the information could be both individual-centric and/or be aggregated. The notion of *complete anonymity* in a pervasive space is that any individual should be indistinguishable from another. If N individuals are associated with the space, an *N-anonymous* solution refers to a solution that affords complete anonymity to each individual. In general, one may seek a k-anonymous solution, where smaller values of k imposes less stringent constraints on the protocol. k-anonymity was first proposed as a measure of degree of privacy for the data publishing problem [20]. The goal there is to publish data in such a way that each individual's record is indistinguishable from at least k-1 other records with regard to the "potentially identifying" attributes. In the current application, we refer to a solution as k-anonymous if it affords an anonymity level of k to all the individuals, i.e., each individual is indistinguishable from at least k-1 other individuals from the perspective of an *adversary*. We call a solution k-anonymous[3] if it satisfies the following criteria.

*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 the individual associated with $e_i$ denoted $e_i.USER$, $\forall i \in [1, n]$ cannot be mapped with certainty to a set of less than k individuals at any $t_j, j \geq i$ (i.e., at no time instant after $e_i$ is generated).

The implication of the above definition on our solution requirements will become clearer in the next section where we describe the adversary's inferencing mechanism. Information about an unknown event $e$ may be indirectly inferred by observing the effects of other events before and after $e$. Therefore, a k-anonymous solution should ensure that no observable property allows the adversary to infer the identity of an individual (associated with any event) down to an unsafe level ( less than k). We now provide an outline the security requirements for our system.

### C. Outline of a Secure Protocol

In our framework, a generic secure protocol to service media events follows the template shown in Fig. 4. Recall that, the SSN is assumed to be the only secure (trusted) component and therefore all communication between the server and a SSN needs to be encrypted. The steps in the event-servicing protocol are the following:

- On event $e$, the SSN sends an encrypted representation of $Enc(e)$ to the server.
- The server on receipt of the query, returns the set of encrypted automaton objects to the SSN that are tagged by $Enc(e)$.
- The SSN decrypts the set of retrieved automatons.
- The SSN advances the state of each automaton if necessary.
- The SSN re-encrypts the automatons (non-deterministically)
- The SSN sends back the re-encrypted automatons to the server. If any automaton reaches its final state, the SSN notifies the server of event detection.

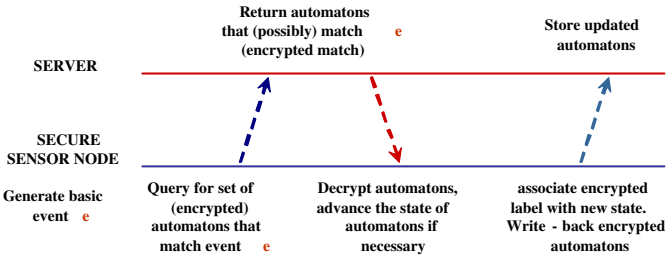


Fig. 4. A generic protocol to service events.

Any secure protocol, denoted $\mathfrak{S}$ can be seen as having the following two components: (i) a secure data model and storage scheme, denoted $\mathfrak{D}$; (ii) a safe suite of communication protocols, denoted $\mathfrak{P}$. These two components in our model are described below.

**Data component $\mathfrak{D}$:** The state information (automatons) are always stored encrypted on the server. The state-table has a fixed number of rows, where each row is mapped to a distinct automaton corresponding to a "rule-individual pair" of interest. We denote the number of rows in the state-table by the constant MAXSIZE. In general the rows-to-automatons map is kept secret from the adversary, but this might not always be necessary as we will see later. Auxiliary indexing information called *tags* are stored along with each encrypted row to enable selective access based on the tag-values. The tags could themselves be encrypted or stored in plaintext. 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 $\mathfrak{P}$:** Let us assume that an encryption scheme exists that allows one to tag each automaton object on the server-side by a set of media events in encrypted format without giving away any information such as "which or how many events are specified in the tag". Fig. 4 shows

[3]The notion of k-anonymity here is not a perfect one since an adversary might be able to establish the identity of the individual associated with an event with a probability greater than the 1/k and this seems unavoidable due to the nature of the application

the generic template of the communication protocol between a SSN and the server in order to service a media event generated by that SSN. For now, we make the assumption that, the automatons 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 [22]).

We will refer to the above description of the system and protocol as the *reference implementation* from here onwards. Our goal is now to derive the set of constraints the two components $\mathfrak{D}$ and $\mathfrak{P}$ need to satisfy in order to meet the k-anonymity criteria (criteria 1). Let us look at a secure solution when the required anonymity level k = N (the total number of distinct individuals in the system).

**A simple N-anonymous solution**: The privacy preserving strategy that satisfies k-anonymity for any $k$ ($< N$) based on the abstract protocol in Fig 4 turns out to be deceptively simple - A simple N-anonymous solution $\mathfrak{S}^*$ = ($\mathfrak{D}^*$, $\mathfrak{P}^*$) is the following.

"*On every event, the SSN retrieves all MAXSIZE automatons from the server one at a time, decrypts it, updates the states of the if it makes a transition on the generated event, re-encrypts the automaton and sends back to the server*[4]."

*Theorem 1:* Solution $\mathfrak{S}^*$ = ($\mathfrak{D}^*$,$\mathfrak{P}^*$) ensures N-anonymity.

**Proof outline**: We see that for all sequence of $n$ events, in the solution $\mathfrak{S}^*$ the protocol $\mathfrak{P}^*$ observed by adversary remains identical. The only information leaked is that $n$ events have occurred in the environment. Therefore no discriminating information is visible to the adversary and the execution is exactly the same for all sequences of $n$ events. Similarly the data component $\mathfrak{D}^*$ seen by the adversary is only a table with a fixed MAXSIZE number of rows. Content of each row is encrypted and on each event, every row gets read and written back (updated). As it is clear, this too does not reveal any further information. Notice that the steps in this solution are sufficient to ensure N-anonymity, though they might not always be necessary.

**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.

This naive protocol tends to become inefficient as the number of rules and individuals grow. To make the solution scalable, a more efficient implementation is required. For this, one needs to tradeoff a certain degree of privacy for enhancing performance. We address these issues in the next couple of sections.

[4]The protocol can be appended with one more step wherein an encrypted message is generated for each row and communicated to a different module which uses the knowledge of "detection of a composite event" to trigger some other action. Of course for security, such a message (whether required or not) has to be generated for each of the MAXSIZE automatons.

## IV. Privacy-Performance Tradeoff

The naive protocol given in the previous section satisfied the k-anonymity criteria for all k $\leq$ N, but from the efficiency point of view, can we do better? It would be ideal if the SSN can retrieve (from the server) only those automatons whose states are affected by the current event. But, this task is made difficult by the fact that state of the automatons and the events that drive them cannot be disclosed to the server. In general, a more efficient protocol tends to recover a smaller set of rows in response to each event (of course this has to be a superset of the set of automatons whose states are affected by the event). We need to devise some secure way to index the encrypted representations of the automatons on the server-side. Then effectively, the server can only observe the rows that are accessed by the SSN during an execution of the protocol and nothing else. (Recall, that the server does not know the correspondence between the rows of the state-table and the automatons.) But, unfortunately, even these observable *row access patterns* can give rise to inference channels that may eventually lead to compromising the privacy of an individual. First, we illustrate the problem of inference with an example.

**Example:** Consider 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 "event servicing" 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. $\diamondsuit$

Inference channels are observable features of $\mathfrak{D}$ or $\mathfrak{P}$ or a combination of both on a sequence of events generated in the space. For a given solution scheme $\mathfrak{S}$ = ($\mathfrak{D}$,$\mathfrak{P}$) and a sequence of events, inference channels can leak sensitive information. Observe that, such 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.

**A note about background knowledge**: As mentioned earlier, we assume that the adversary knows the set of individuals who are associated with the space. He also knows what rules apply to which individuals the structure of the automatons corresponding to each rule. Besides this, we assume that the adversary cannot exploit any alternate channels like "the time interval between two media events", "the time of an event" etc. to deduce auxiliary information. For example, we will assume that by simply observing the time difference between two media events generated by the pervasive space, he is not able deduce whether these two events are in the same physical location or not.

### A. Inference via Access Patterns

In general, an event $e$ may be identified by determining one or more of the automatons 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. The small example below illustrates what comprises an unique signature of an automaton.

**Example**: Let there be a set of 5 automatons $\{A^*, A_1, A_2, A_3, A_4\}$ that make a transition on event $e_1$. Let there be another set of 5 automatons $\{A^*, A_5, A_6, A_7, A_8\}$ 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 automatons. Now, if the adversary sees a row (in the row-encrypted state table) that is accessed along with 2 distinct sets of 4 automatons (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. $\diamondsuit$

Recall that the adversary knows the details of the automaton that are being implemented in the system and their characteristic patterns as well (which can be pre-computed) in the form of back-ground knowledge. As illustrated in the example above, the characteristic access patterns of an automaton might still be discernible after encryption of the rows and events. In other words, "*a simple anonymization scheme that hides only the explicitly identifying information does not guarantee the removal of such access-pattern based inference channels*". Now, we formally define the notion of access patterns and provide theorems characterizing the nature of inference channels.

*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 automatons, 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 $x_3 \leftarrow F^{'''}$ assuming a suitable sequence of events exists which generates this pattern of automaton access. In general there maybe multiple patterns following the same template. Consider the following example.

**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 5 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. $\diamondsuit$

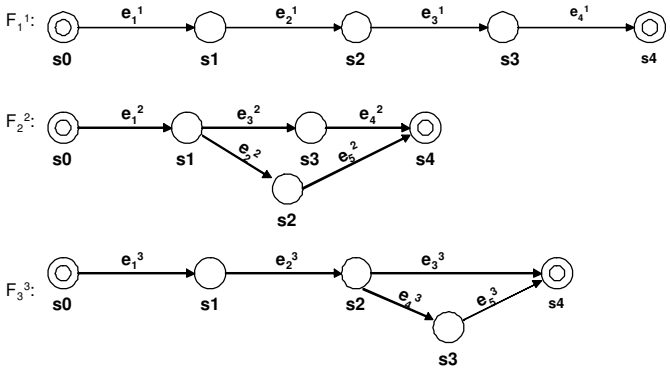


Fig. 5. Example - Automatons and event sequences.

Now, we define a *characteristic pattern* of an automaton 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. Depending on the mapping of rules to individuals and the structure of the automatons that implement these rules, the *complete set of 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 $\mathbf{CP}(\mathbf{F})$.

**Example:** The figure below shows 3 automatons (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 automatons. In general the

characteristic set of an automaton (as well as some subsets of it) could be unique.♢

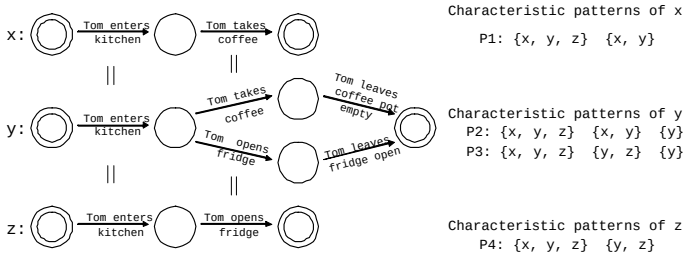


Fig. 6. Characteristic patterns.

**Pattern Analysis by an Unbounded 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 automatons and thereby determine the identity of the individuals. Note that, even if a rule $R$ applies to all individuals in the space, for a given individual $I_1$, depending on the set of other rules that apply to individual $I_1$ the automaton $F_R^{I_1}$ might generate completely different set of characteristic patterns than the automaton $F_R^{I_2}$ for another individual $I_2$.

*2) Observable indistinguishability:* As we noted above, the characteristic set of an automaton could be unique. Therefore two automatons are said to be *observably indistinguishable* or simply indistinguishable if their characteristic sets are *identical*. We define the notion of *pattern isomorphism* (simply isomorphism for short) between two sets of automatons $\mathcal{A}$ and $\mathcal{A}^{'}$ where $|\mathcal{A}| = |\mathcal{A}^{'}|$ as follows.

**Definition** *4:* **(Pattern Isomorphism)** Let $G : \mathcal{A} \rightarrow \mathcal{A}^{'}$ be a bijective (i.e., 1-1 and onto) map from $\mathcal{A}$ to $\mathcal{A}^{'}$. Then $G$ is called a pattern isomorphism iff for all automatons $a \in \mathcal{A}$ 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) for all 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 automatons, 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 automatons onto itself), then each automaton and its image under this map (also an automaton in $\mathcal{A}$) will be observably indistinguishable to the adversary in our model. 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 automatons. We will call such automorphism (if it exists), a *non-identifying automorphism*.

We now look at the necessary and sufficient conditions for achieving k-anonymity in our model using the concept of observable indistinguishability.

*3) K-anonymity in terms of patterns:* The adversary is able to see the pattern of row-accesses from the state table corresponding to a sequence of events. As a result he can extract the embedded characteristic patterns of rows from such a pattern. E.g., from the observed row access pattern $\{\{r_1, r_2\}, \{r_1\}, \{r_2\}, \{r_1\}\}$, the adversary can easily determine the embedded characteristic pattern for $r_1$ as $\{\{r_1, r_2\}, \{r_1\}, \{r_1\}\}$ and that of $r_2$ as $\{\{r_1, r_2\}, \{r_2\}\}$. The adversary's goal is to determine the identity of the individual involved in each of the media-events in this sequence. Consider a particular media event $e$. The adversary can only infer the identity of $e.USER$ indirectly by associating the set of automatons that were affected (retrieved from the state table) while servicing $e$. Since our basic assumption is that the adversary does not know the exact mapping of rows to the set of automatons, he can only try to identify the automatons in a row via the observed characteristic pattern of the row. Now, we state the necessary and sufficient condition for k-anonymity (one that ensures that criteria 1 is satisfied), but first we define the term *diversity* associated with a set of automatons as follows:

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

*Theorem 2:* **(k-anonymity: Necessary & Sufficient condition):** Given an injective map (assignment) $G_0 : \mathcal{A} \rightarrow R$ from set of automatons 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 automatons-to-rows 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 automatons (which are accessed on $e_i$). Since the only observable features about the encrypted automatons 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 automatons up to which an encrypted automaton can possibly be identified is at least $k$".

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

From the adversary's viewpoint, the $k$ sequences ($E_0$, ..., $E_{k-1}$) are indistinguishable. In other words, for each of these $k$ sequences if there exists an assignment (map) of automatons to rows such that for all rows in the table, the $k$ automatons 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 automatons. $\diamondsuit$

The problem of detecting whether a given set of automatons 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 3:* **(Hardness of checking for k-anonymity):** The problem of recognizing whether a given set of automatons (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 automatons". The reduction is presented in appendix . $\diamondsuit$

Now, consider the problem of designing an efficient 2-anonymous solution for a given set of automatons. If the set of automatons are naturally 2-anonymous (i.e., a suitable fixed-point free pattern automorphism on the set of automatons exists), then we could simply use an encrypted-matching based automaton access protocol as illustrated in figure 4 in the previous section and be assured that the execution cost of the protocol is minimized[7]. But the problem is that such a verification of the 2-anonymity criteria is prohibitively expensive (NP-Complete) and therefore not possible in the general case. Under such circumstances, we have to design an alternative solution scheme that can guarantee the required level of anonymity. We also need to ensure that the correct automaton(s) are retrieved on every event. This leads to a constrained optimization problem where the objective is to minimize the overhead (in terms of false-positive retrievals) while ensuring that access patterns are safe. Next, we present a solution scheme and describe the space it searches for the optimal solution.

### B. Anonymization via Event Generalization

*Generalization* is a natural technique for achieving k-anonymity in a variety of applications, and has been widely used for the privacy-preserving data publishing applications [20], [23]. Generalization, basically refers to the process of clustering (or partitioning) a set of items into groups of size at least k and replacing every element's identifying information by the more generalized information at the corresponding cluster-level. While this process induces indistinguishability between elements (within a cluster), it also leads to information loss. In data publishing, the information-loss is measured with respect to the intended utility of the data, e.g., a suite of data mining algorithms. Analogously, in data-querying/data-retrieval systems, the information-loss is an inverse measure of the performance. For instance, it may denote the average number of false-positives retrieved on a query [8]. From the foregoing discussion, a natural question that arises is whether a generalization-based approach can achieve k-anonymity for our application? Such a solution will need to suitably obfuscate the information leaked via access patterns of automatons.

In our model, the set of all media events generated by the SSNs is finite and enumerable (it is simply the cartesian product of the set of all *individuals*, *spatial-regions*, *resources* and *actions*). In fact, we are only interested in the subset of media events that can cause a transition in some rule automaton. Under the generalization model, a candidate solution scheme is specified as $\mathfrak{S}$ = ($\mathfrak{D}$, $\mathfrak{P}$, $\mathfrak{C}$) where $\mathfrak{D}$ and $\mathfrak{P}$ are same as before and $\mathfrak{C}$ is an event generalization scheme. Now, consider the following generalization scheme:

*Solution 1:* **(k-Individuals Clustering)** 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 implies a *static* indexing scheme for the automatons (rows in the state-table), i.e., the index-tag does not change with the state of the automaton. It is easy to see that solution 1 leads to a protocol where the SSN always requests the server simultaneously for all automatons associated with the k or more people assigned to a cluster. This obviously leads to a homogenous access pattern for all of these automatons, thereby leaving no inference channel open for the adversary to determine which particular automaton within a cluster has been fired or equivalently which individual might have triggered the media event. Therefore, this solution satisfies the k-anonymity criteria. Next, we show that solution 1 can be significantly improved upon in terms of performance. The improved solution is based on the notion of a *connected group*.

**Definition** *6:* **(Connected Group)** The connected groups of automatons for an individual $I$ correspond to the set of connected components in an undirected graph with vertex set as the set of all automatons of $I$ and edges corresponding to every pair ($F_x^I, F_y^I$) of automatons (where $x$ and $y$ are two distinct rules) such that both automatons can make at least one transition between states on a common event $e$.

**Example:** Fig. 7 shows two connected components corresponding to TOM. All the 3 automatons in the first group can make a transition on the event "TOM, KITCHEN, *, ENTRY" and the second component corresponds to the automatons with the common edge "TOM, SERVER_ROOM, *, ENTRY".

As can be seen from the example, a connected-group represents a single individual, but there may be multiple connected-groups corresponding to an individual. Now, a more efficient solution $\mathfrak{S}$ = ($\mathfrak{D}$, $\mathfrak{P}$, $\mathfrak{C}$) is one where $\mathfrak{C}$ is given by the following.

*Solution 2:* **(k-connected-group Clustering)** Partition the

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

[7]Since exact matching is assumed to be possible over the encrypted representations, hence no false-positives are retrieved from the server

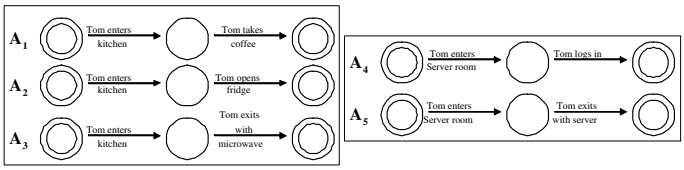


Fig. 7. Example: Connected Groups

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

*Theorem 4:* **(Sufficient condition for k-anonymity)** Solution 2 is k-anonymous.

**Proof:** The clustering scheme proposed above creates a partitioning of the set of automatons such that automatons belonging to different partitions (clusters) are never accessed simultaneously on any event, whereas all automatons 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 automatons belonging to a cluster). Therefore, the characteristic patterns for all automatons (rows) within a cluster are the same making them observably indistinguishable from each other. E.g., If the events of automatons, $A_1, \ldots, A_m$ (say, corresponding to rows $r_1, \ldots, r_m$ in $DB$) are clustered 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 cluster can be seen as a generator of one such class of patterns that is characteristic of every automaton in that cluster. 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 2. $\diamondsuit$

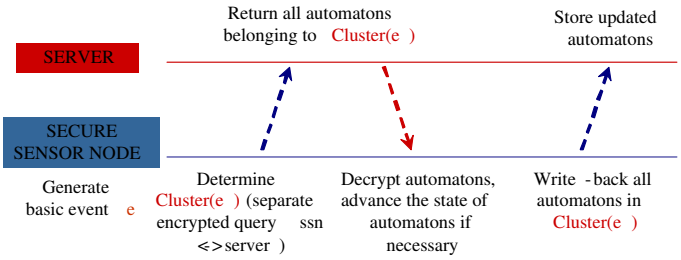


Fig. 8. Secure protocol for servicing events.

Now, the resulting k-anonymous protocol (for both, solution 1 and 2) is shown in figure 8. Note, the NP-Completeness result makes it computationally infeasible to design a more efficient solution (that is guaranteed to be k-anonymous) in the general case (i.e., automaton structure may be arbitrary directed graphs with cycles). For instance, consider the following event-partitioning 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 allows for a larger (superset) of partitioning schemes with larger number of partitions. The question is whether such a k-diverse event-clustering scheme achieves k-anonymity since each individual media event is not identified beyond the cluster-id. The answer is no! Such a clustering scheme transforms the set of automatons by effectively adding many self-transitions to the states (nodes) and does not change the complexity of the problem of "checking for anonymity condition" as such. In appendix we illustrate with an example how in general, a k-diverse clustering scheme need not achieve k-anonymity under these conditions. The reason being, the characteristic set of the modified automatons may still be unique and therefore allow an adversary to infer its true identity by just observing row-access patterns in our model. (For security analysis, we need to assume that the event clustering scheme is public knowledge (i.e., 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.

In the next section, we present an algorithm for computing "low cost" partitioning of the automatons into clusters such that average number of false-positive retrievals (per event) is minimized.

## V. Clustering Scheme

The solutions proposed in the previous section poses a natural optimization problem: since different automatons may be accessed at differing frequencies, some clustering schemes will have lower average *costs* than others. Therefore, the goal is to partition connected groups of automatons in order to minimize this cost while ensuring that 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 generalization scheme). Our solution uses the following observation

**Observation** *1:* **(Flexible anonymity)** The k-anonymity criteria only stipulates that each cluster of automatons have a diversity of k or more and it is not required that all automatons corresponding to a single individual belong to a single cluster.

Thus, the clustering scheme can assign two connected groups of automatons corresponding to the same individual to two distinct clusters[8]. 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

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

cost (more efficient) solutions. In the remainder of the section, first we will model the above problem as a novel set partitioning problem with some unique constraints. Then we will show that this problem is NP-hard and propose a heuristic algorithm that provides good practical solutions.

### A. A Partitioning Problem

Let us assign each individual a distinct *color* (N colors in all). 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 (i.e. number of events that can cause the automaton to make a transition). 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 one (a single unit) 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. One may use various measures for the weight and employ any heuristic or domain-specific knowledge. We assign the weight of an automaton to be equal to the number of edges in its directed graph. We assume that the number of edges reflects the probability with which an automaton might be accessed on a randomly generated media-event in the pervasive space.

In the pre-processing phase, the connected components are computed as follows: all automatons of the same color that have at least one edge in common are fused into a new "larger" ball of same color but with larger weight and price. Both the weight and price of the resulting 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 respectively (i.e., the probability of access of the connected group is equal to the sum of access-probabilities of the component automatons, as is the transmission overhead). Now, the optimization problem becomes the following.

*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.

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

**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 [5] which is a known NP-hard problem, can be reduced to an instance of our balls-and-bins optimization problem. (Details are relegated to the longer version of the paper). We also note that, the more constrained case, where all automatons 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, automatons and individuals.

The input to the problem are 3 arrays of size equal to the number of balls $|B|$. The first array $Color[1 \ldots |B|]$ specifies the color of each ball, the $2^{nd}$ array $Weight[1 \ldots |B|]$ specifies the weight of a ball and the last array $Price[1 \ldots |B|]$ specifies the price of each ball. We also specify the number of distinct colors or individuals in the system as NUM_COLORS (which is same as N) and the required anonymity level $k$. The output is the cost of the binning scheme and an array $Bins$ also 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 pseudo-code is given in the Algorithm. 1.

Note that in the above algorithm, the candidate ball-transfers and their associated cost reductions can be modelled 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 [24]) can be utilized for computing the **most profitable** matching in the graph[9]. Ball-exchanges between bins can also be modelled similarly. Finally, it can be seen that the algorithm terminates by noting that the last *while* loop in figure 1 will definitely terminate since only positive cost-reductions are allowed.

## VI. Implementation & Evaluation

We built a prototype of the system described in this paper which extends the system presented earlier in [26]. In [26] the goal was to build an end-to-end framework for privacy-preserving data collection that focused on particular modalities for data collection (namely video and RFID). The focus there

[9] In our implementation we also allow "ball-exchange" between bins. 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.

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

was on instrumenting the pervasive space with sensors to detect media events and to implement simple (instantaneous) rules that did not require prior state information. For this work, we extended the system in [26] to support the architecture presented in this paper and integrated the new algorithms with it. Below, we give a summary description of a "proof-of-concepts" deployment we did with a few hand-picked rules defined with respect to our office settings. We present some preliminary results to (i) see estimated benefit from using our clustering algorithm and (ii) gain an insight into the characteristics of load (security-scalability trade-off) such a system needs to bear with varying parameters, e.g., number of individuals, number of rules, degree of anonymity etc.

### *A. A "Proof of Concept" Application*

We model 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[10] 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 broken 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 automatons that would implement these rules (i.e., detect the corresponding composite events). All 5 automatons 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 and "entry into the kitchen" event. As a result, all the automatons 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.
3) An individual enters the server-room and moves the SUN server.

[10]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.

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 automatons 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.

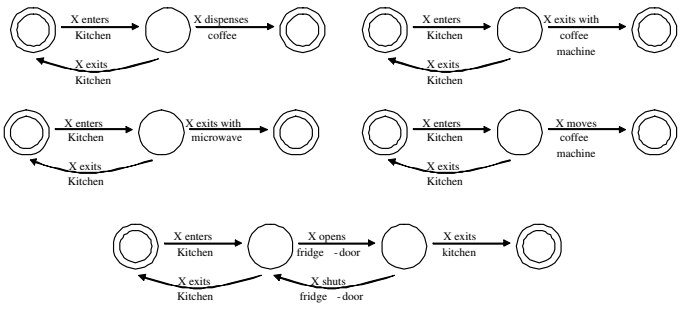


Fig. 9. Kitchen rules

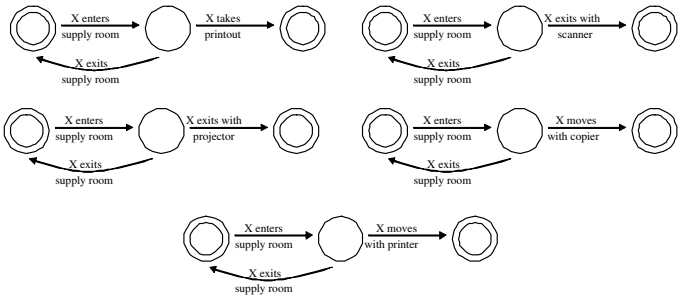


Fig. 10. Supply-room rules

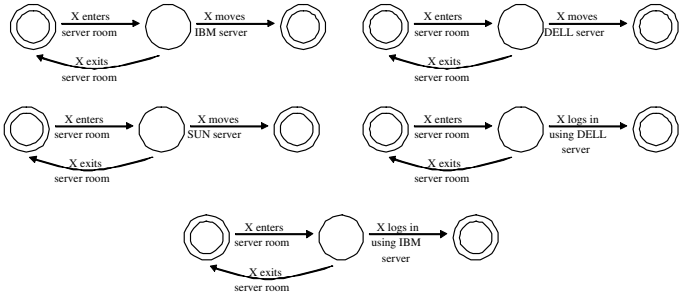


Fig. 11. Server-room rules

Next, we present a summary of the performance characteristics of the clustering algorithms and other load characteristics of our system from experiments using the above data set.

**Performance of clustering algorithm:** We compare the performance of two solutions schemes: (i) *k-Individuals Clustering* and (ii) *k-connected-group Clustering*. We use the clustering algorithm described in the previous section (Algorithm 1) to compute the optimal solution in both the cases. We then compare the performance characteristics predicted by the algorithm (i.e., cost estimates) with the actual number of automatons retrieved in servicing a sequence of randomly generated 1000 events.

**Generation of event sequence for simulation**: The weight associated with each automaton 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 interrelated, it is very difficult to compute in a simulation. Therefore, we simply associated 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 automatons 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. Recall, our cost measure is:

$$\sum_{B \in Bins} \Big( \sum_{ball \in B} weight(ball) \Big) \times \Big( \sum_{ball \in B} price(ball) \Big)$$

Fig. 12 shows the **estimated costs** (given in previous section) of the 2 partitioning schemes for various values of the parameters $k$ (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 automatons 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 $k$.

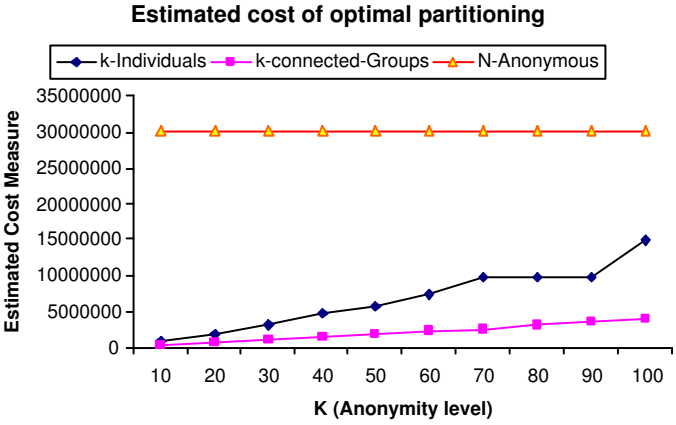


Fig. 12. Cost estimates for the constrained ("Together") and unconstrained ("Separate") partitioning schemes

For the second plot, we generated a sequence of 1000 media events as described above. Fig. 13 shows load characteristics of the two clustering schemes on this test sequence. The y-axis is the total number of automatons retrieved over the 1000 events and therefore represents the transmission-load due to the "SSN-server communication" in a real setting.

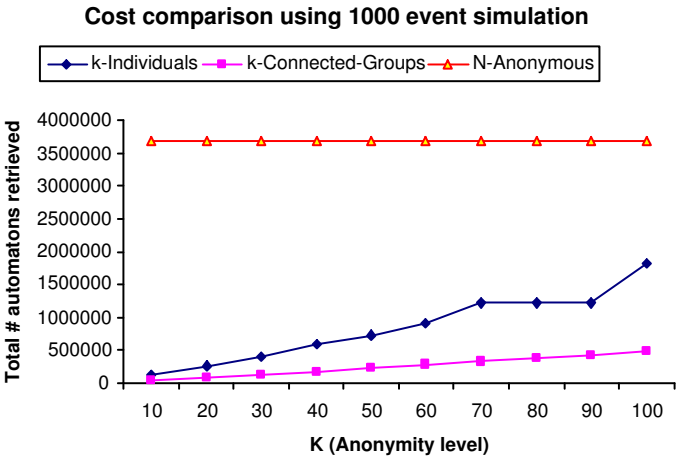


Fig. 13. Total # automata retrieved in 1000 events

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 Clustering* based anonymization approach as compared to the *k-connected-group Clustering* 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.

**Experiments using Synthetic Data**: In addition to the above experiments, we performed others using synthetic data that was generated by varying various parameters like average "connectivity" between rules, number of individuals, number of distinct rules etc. We describe the process of synthetic data generation below.

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

**Cost estimates with varying parameters**: 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 automatons in the state-table. The plot shows that the cost differential is expected to increase consistently with increasing number of individuals and automatons in the state-table, but shows no clear trend across varying values of k.

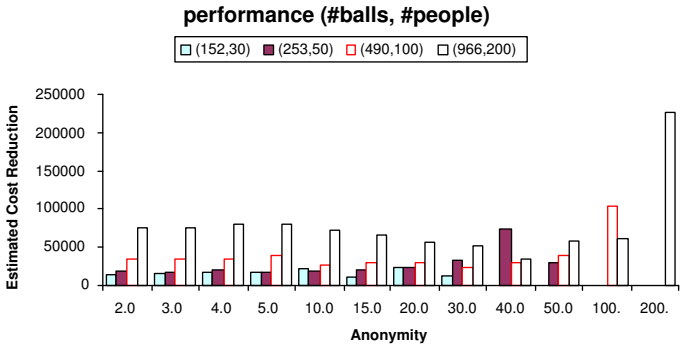


Fig. 14. Analytical estimate of performance improvement

## 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 [22] 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 look-up table (read-only) subsequently. The first column of $T_{EC}$ stores $E_{k_{SSN}}(\text{“}e_i^j\text{”})$ 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 $E_{k_{SSN}}(\text{“}e_x^y\text{”})$ 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 automatons 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 automatons at the same time (since automatons 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[11], say $SSN_i$ and generates a secret key $s_k$ for encrypting the automatons.
- 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 (automatons) 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 $0.5$.

## VII. Related Work

There is a large body of work in the area of systems for pervasive computing environments [14]. GAIA [19] is an example of a system designed inherently to enable pervasive computing. Other systems such as CRICKET [18], 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 [2]. Langheinrich [12] 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 [1] combines hop-to-hop routing based on handles with limited public-key cryptography to preserve privacy from eavesdroppers and traffic analyzers. Our previous work [26] 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) [15] 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 [6], [15] 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 [4].

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 [20], [23]. More recently, some work on location-privacy also incorporate similar measures for privacy [7], 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 [16] which is of interest since we too model the pervasive space as a stream of events. The authors in [16] 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 [16] 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 [16].

## VIII. 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

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

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 [1] 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] 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, 2002.

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

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

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

[5] 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.

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

[7] B. Gedik, L. Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model In *ICDCS*, 2005.

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

[9] B. Hore, J. Wickramasuriya, S. Mehrotra, and N. Venkatasubramanian. Privacy-preserving event detection for pervasive spaces. In *UCI-ICS technical report*, 2007. http://www.ics.uci.edu/∼bhore

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

[11] 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.

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

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

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

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

[16] R. Ostrovsky, and W. E. Skeith III Private searching on streaming data. In CRYPTO, volume 3621 of LNCS, pages 223–240. Springer, 2005.

[17] 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)*, 2003.

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

[19] M. Roman and R. H. Campbell. A Middleware-Based Application Framework for Active Space Applications. In *Middleware*, 2003.

[20] P. Samarati and L. Sweeney. Generalizing Data to Provide Anonymity when Disclosing Information (Abstract). In *PODS*, 1998

[21] 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.

[22] 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.

[23] L. Sweeney. k-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10 (5), 2002; 557-570

[24] 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.

[25] 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,*LNCS*, pages 201–212, 2004.

[26] 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

**Example:** 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. 5 (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. 15 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).

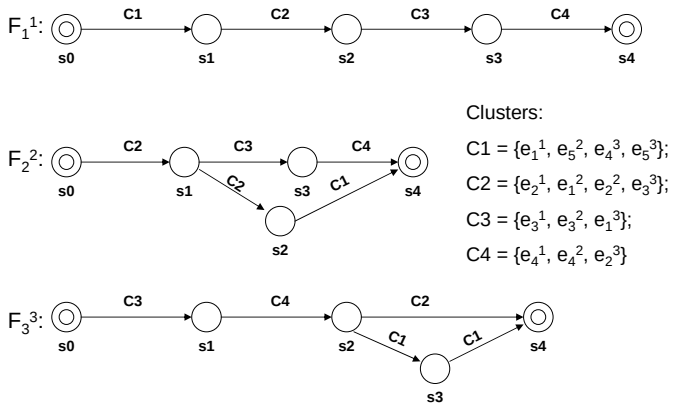


Fig. 15. Example - Automatons indexed by event clusters.

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.) $\diamondsuit$

Here we present a proof of theorem 3 from section IV. 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-B), 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 [13].

**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** *2:* 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. $\diamondsuit$

Fig. 16 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.

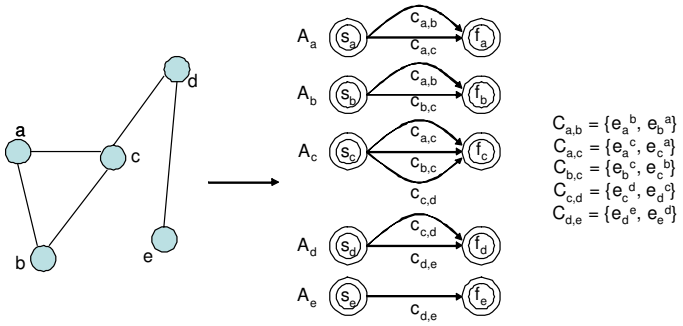


Fig. 16. Example: NP-Hardness reduction

We show that the minimum cost ball partitioning problem of section V is NP-hard by reduction from the "minimum sum-of-squares" problem [5]. 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. $\diamondsuit$