LAD: Learning Access Control Polices and Detecting Access Anomalies in Smart Environments

Tomasz Kalbarczyk, Chenguang Liu, Jie Hua, Christine Julien
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX, USA
{tkalbar, liuchg, mich94hj, c.julien}@utexas.edu

***Abstract*—The domain of access control has long suffered from a lack of expressiveness in specifying access control policies. Recent approaches have leveraged contextual fingerprinting to formulate access control frameworks for both generating and enforcing access control policies. However, effectively and automatically identifying the context attributes relevant for access has proven challenging and cumbersome. An approach that shows promise in supporting more expressive and easy-to-use attribute-based access control relies on recent advances in continuous neighbor discovery protocols and low cost wireless communication technologies such as Bluetooth Low Energy (BLE). These technologies have created opportunities to build smart environments that can seamlessly and inexpensively provide rich contextual data. These capabilities have the potential to enable new transparent and automatic approaches to defining and evaluating access control policies for mobile users and for detecting anomalous access patterns in smart environments. In this paper, we present the LAD framework that uses raw contextual data available via technologies such as BLE to derive real-time *attributes* defined by the presence of mobile and static nodes in the nearby environment. Based on user interactions in these environments, our framework learns appropriate access control policies and enforces these policies based on attributes that change in real-time as users move in the smart environment.**

***Index Terms*—access control, pervasive computing, smart environments, anomaly detection**

I. Introduction

Our everyday environments are increasingly "smart". In this new world, from controlling physical door locks to browsing virtual data that is pervasively available via cloud connections, our mobile devices serve as gateways for access to both physical and virtual resources. Advances in low-power wireless communication and cloud computing are enabling flexible, dynamic, and seamless access to these resources. However, these new environments come with copious new security challenges. Mobile devices increasingly represent single points of failure as the primary user interaction points into these systems. If attackers can gain unauthorized access to a user's device, they have broad access to a multitude of systems. Work in context-aware computing [1]–[3] has focused on protecting access to personal mobile devices using on-device characteristics and measurements from on-device sensors. While some of these approaches use contextual data from the environment to protect access or to authenticate users, they focus solely on controlling access to a given user's personal mobile device. The problem of ensuring that the users (and their devices) have access to the appropriate shared resources within the "smart" environment remains.

While the field of access control is well-established, these new shared and collaborative environments demand new access control paradigms [4] that are less rigid than traditional approaches that rely exclusively on user roles or identities [5] placing undue burden on system administrators to assign the appropriate roles and identities. Attribute-based access control (ABAC) [6] is a promising model that relieves the burden on the system administrator and allows for dynamic and flexible access control policy specifications. At a high-level, users are assigned *attributes*, and resources are secured via policies that reference these attributes. If a user's attributes satisfy a given policy, the user is granted access. ABAC lends itself to generating the expressive, fine-grained access control policies that are necessary in dynamic "smart" environments where access control decisions could benefit from considering a wide range of environmental contexts. However, determining how to generate user attributes remains a challenge. Work in tag-based access control [7] leverages user-generated tags associated with data to create and assign attributes that prove useful for access control. However, this work still relies on manual tagging from users, which inherently limits the granularity of access control rules that can be expressed via these attributes.

In this paper, we explore automatically generating attributes using *context* information that users' devices transparently collect from the smart environment. Recent advances in continuous neighbor discovery [8]–[11] that leverage communication protocols such as Bluetooth Low Energy (BLE) have made it feasible for mobile applications running on user devices to have real-time access to contextual data that reaches beyond the capabilities of the user device. This real-time contextual data includes information regarding proximity to other user devices, static devices in the environment, and sensor readings [12], [13].

Our LAD framework captures contextual data available in the proximal environment from which it seamlessly and automatically generates explicit spatiotemporal, context-based *attributes*. LAD derives attributes via BLE beacons that a user's mobile device receives from inexpensive, low-powered static devices built into the environment and from mobile

devices carried by other users, though conceptually LAD is generalizable to signals received on any wireless interface. As users move and spend time in the environment, their devices receive beacons, from which LAD derives attributes. We refer to these attributes as *spatiotemporal attributes* because they capture the relationship between the user and the space the user occupies over time.

In LAD, the user's device stores all received beacons in a local beacon repository. In the simplest sense, a user's device captures an attribute just based on having been in the presence of a specific beacon (e.g., the beacon near a building's main entrance). Alternatively, the user's device can compute more complex attributes. For instance, to capture an attribute derived from the presence of another user's mobile device, the current user's device may need to receive multiple beacons from the other user's device for an extended period of time [14].

Over time, a user's attribute set can grow quite large, however not all of the captured attributes are relevant at a given time, a given place, or to a given access control challenge. Further, a user's attribute set might itself be quite revealing, from a personal privacy perspective. For these reasons, at the time of an access control request, a user device can automatically *filter* the set of attributes based on either space, time, or both. In addition, the resource to which access is being requested may have its own interests in terms of attributes relevant to the access control decision. Conceptually, the resource also provides a filter on the attribute set. Together, these filters over a user device's attribute set define a *partition* of attributes over which an access control policy will be evaluated.

Consider a scenario in which a user, Carla, moves from the main entrance of a smart building to a lab on a upper floor of the building. In doing so, Carla's device *captures* a sequence of beacons from the main entrance, through the elevator lobbies and down a hallway on the upper floor. As she arrives in front of the room's locked door, the resource (i.e., the smart lock that secures the room) must either grant Carla access to the locked room or deny access. The decision can be based on the spatiotemporal history captured in Carla's attribute set. Further, the resource (i.e., the lock) can use additional information as part of the decision process. For instance, if the current time is within the building's operating hours, the access decision might be granted based on Carla's captured attributes. If it is after hours, the request might be denied. However, if it is after hours and Carla's device has captured the beacon of an authorized person's device within the very near past (e.g., 2 seconds), the request might be granted, even after hours. Note that, to navigate this access control decision, Carla was able to be completely passive and let her movements in the smart space do all of the work.

A key contribution of our approach is the automated derivation of spatiotemporal attributes. However, to allow Carla access to the door, we also need a way to generate a rule or set of rules against which to verify her *attributes*. Similar to existing work in generating access control rules [1], [15], we assume a training period during which users must use manual access to the resources. This training period allows us to gather a representative set of access logs correlated with attributes. That is, for some period of time, Carla's device must capture *attributes* and provide *partitions*, while she manually unlocks the door to the secured lab.

We utilize work in association rule mining (ARM) [16]–[18] as a basis for generating frequent item sets from our logs. These are subsets of attributes that occur together within some threshold of frequency. We devise a parameterized, rule mining algorithm that uses these item sets to iteratively generate a representative set of rules for access control. The primary parameters to our algorithm consist of: the length of an itemset in terms of the number of *attributes*, the coverage of the itemset in terms of the number of *partitions* it covers from our logs, and the similarity of an itemset to existing rules.

For a given resource, the LAD framework executes this rule mining algorithm across a range of hyperparameters to generate a candidate set of policies (rule sets) that govern access to the resource. Each rule set has varying performance in terms of its true positive rate (TPR) and false positive rate (FPR). Certain access control use cases might be more sensitive to TPR or FPR. Providing a candidate set of policies makes our framework tunable to the rigor of the access control use case and/or the requirements laid out by a system administrator.

Our evaluation demonstrates that, in terms of access control accuracy measured by TPR and FPR, our algorithm out performs existing state of the art, binary classification machine learning approaches such as extreme gradient boosting (XGB) and random forests (RF). Moreover, our approach benefits from generating human interpretable rules. To our knowledge, our framework is the first to derive explicit attributes from a real-life sensor testbed deployed in an industrial-sized, multi-story, large occupancy building, to use these attributes to generate access control logs, and to learn access control policies or detect access control anomalies based on application requirements.

Our concrete contributions are the following:

- We transparently generate spatiotemporal attributes from raw contextual data received from both static and mobile devices in the environment.
- We use context-derived attributes to automatically learn access control rules for both enforcing access control and detecting access anomalies.
- We create a novel rule-mining algorithm that iteratively generates human-interpretable access control rules.
- We deploy an extensive sensor testbed across 4 floors of an engineering building, including 24 static devices and 7 mobile devices each carried by a participant recruited to capture data over the course of 45 days.
- We evaluate our approach against two other machine learning approaches (XGB and RF) based on the data collected from the testbed.

In the next section, we present the related work, section III discusses the details of the LAD framework, and section IV demonstrates the performance of our framework in terms of access control accuracy.

II. Background and Related Works

The LAD framework lies at the intersection of several fields, including access control, trajectory mining, anomaly detection and association rule mining. In this section, we review the related works in each of these fields as they pertain to LAD, highlight the current gaps in the research, and show how LAD addresses these gaps.

Attribute-based and Context-aware Access Control. The closest related work to LAD is in the field of ABAC [6], [7], [19] and context-aware access control [1], [20] for mobile devices. Most of the works in the former rely on user-defined policies and attributes, which must be specified manually. LAD provides automatically generated attributes and policies. Existing work in mining access logs to generate ABAC rules [15], [21] focuses on generating comprehensive, strictly satisfied rule sets over strictly defined synthetic attributes. Our attributes are derived from real-world raw contextual data that is ever-changing and inherently noisy. These types of attributes necessitate a flexible rule mining algorithm that is probabilistic in nature to accommodate changes and noise in the environment. Existing approaches impose strict constraints on rule generation that are not suitable to the types of environmentally-derived attributes that we employ.

Works in context-aware access control focus on protecting access depending on the status of device sensors, and sensors in the environment. For example, ConXsense [1] uses the notion of context familiarity based on recurring presence of bluetooth signals to designate an environment as "safe" or "unsafe" for device use. These approaches derive features from the environment, and apply machine learning algorithms to automatically make access control decisions. Much like LAD, they automate the binary classification task of providing access control. These approaches only focus on access to a user's mobile device, and necessitate user feedback regarding when the environment is not safe. In other words, they require negative, supervised feedback, which is not realistic for most access control use cases, where intruders are rare. LAD is designed to automatically provide access control in scenarios where negative feedback is sparse or entirely unavailable.

Trajectory Mining. The *attribute* sets derived via the LAD framework share similarities to temporal trajectories. The field of trajectory mining is burgeoning due to the recent explosion of user location and check-in data. Most of these approaches [22]–[24] focus on predicting future user movement patterns based on social media feeds, like Facebook and Twitter check-ins. However, these approaches are focused on massive datasets at the "city" scale. Meanwhile, LAD targets the "building" scale, where datasets are smaller and providing pin precise localization is challenging and expensive. Accurate indoor localization solutions necessitate extensive environmental tuning [25], [26], are sensitive to changes in the environment [27] and/or require the use of expensive hardware such as cameras [28], [29]. LAD performs at the "building" scale, uses cheap, off the shelf environmental sensors and does not necessitate sensors that are finely calibrated to the environment.

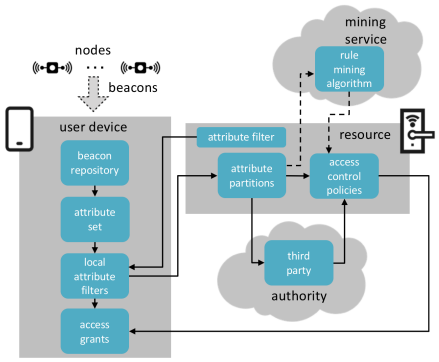


Fig. 1: LAD Conceptual Architecture and Key Components

Anomaly Detection Lots of work has been done focusing solely on trajectory-based anomaly detection at the "building" scale. Most of this work comes from the computer vision community and focuses on using cameras to track users in the environment [28], [30]. Our approach does not actively monitor or track users, since we use passive sensors in the environment that transmit beacons. The user devices capture the attributes and only reveal the attributes that they want to reveal at the time of access.

Association Rule Mining The rule mining algorithm at the heart of LAD leverages long-standing advances in the field of association rule mining [16]–[18]. Specifically, we make use of the FP-growth ARM algorithm to efficiently generate sets of potential rules. Recent work in trajectory-mining has also leveraged ARM [31] for mining trajectories using grids. This work splits the environment into grids based on localization data. As mentioned earlier, LAD does not require precise localization.

III. LAD Framework

At a high level, the LAD framework comprises five logical components that interact with one another, as shown in Fig. 1: (1) *nodes* are static and mobile devices that are capable of transmitting periodic wireless beacons; (2) a *user device* receives wireless beacons from the *nodes*, constructs a user's personalized captured *attributes* for use in access control and anomaly detection, and may serve as a *node* itself; (3) a *resource* is a physical or virtual object to which the user (or the user device) wants access; (4) the *authority* validates the authenticity of *attributes* presented by the *user device*; and (5) the *mining service* uses access logs and provided attributes to learn access control rules for the resource.

A. An Overview of LAD

The primary goal of our framework is to manage access to a given *resource* or set of *resources*. We define a *resource* to be any physical or virtual object to which access can be granted or not granted based on a physical or virtual barrier imposed by a system administrator. Our evaluation considers access to physical resources, such as elevators, rooms, drawers, computers, or lab equipment; however, our approach is also

directly applicable to virtual resources such as files stored at the cloud or on edge devices.

We use Fig. 1, which depicts the conceptual architecture of our framework to help describe the necessary steps for a user device to gain access to a specific *resource*. First, *nodes* in the environment transmit beacons that are received at the user device. These beacons are processed and stored in the device's beacon repository; they are then captured and stored as *attributes* in the device's attribute set. Since only a subset of user *attributes* are relevant to a specific *resource*, whenever a user device requests access to a *resource*, as part of the handshake, the *resource* provides a filter to the user device. The user device filters its set of *attributes* based on this filter and its own local filter for releasing its attributes and generates a *partition* that is specific to a particular access request. To complete the handshake with the *resource*, the user device provides the *partition* for validation. For now let us assume that we already have an access control policy. The *resource* first validates the *attributes* in the *partition* with the *authority*, then the *resource* checks whether the *attributes* satisfy the *resource*'s access control policy. If satisfied, the user device is granted access to the *resource*. This entire process is depicted in Fig. 1 using solid arrows. Periodically (e.g., according to the system administrator), LAD mines access control rules from the provided attribute partitions and generates access control policies; this process is shown via the dashed arrows in Fig. 1.

B. Access Control Policies in LAD

Before delving into the details of how *attributes* are captured, *partitions* are constructed, and access control policies are learned, we next define how policies are represented

An access control rule is logically represented as a conjunctive clause where each literal denotes an *attribute*. As such, a rule is satisfied if all of its *attributes* exist in the *partition* provided by a user device. A access control policy is logically represented as a disjunction of rules, meaning that in terms of *attributes* each access control policy is logically represented in disjunctive normal form (DNF). If any one of the rules is satisfied, then the access control policy is satisfied, and the user is granted access to the *resource*. Note that this rigid structure may seem restrictive, but any boolean formula over attributes can be rewritten in DNF. The structured nature of this approach allows LAD to learn each of the clauses independently, with the assistance of the rule mining service.

C. Use Cases of LAD

We next provide a series of access control and anomaly detection use cases made possible by the LAD framework. These example use cases differ in the importance they place on false positives and false negatives. A false positive *over provisions* access, granting access to a user who should not have been granted access. If the resources are particularly sensitive, false positives can be completely unacceptable. A false negative, in contrast, *under provisions* access, preventing a user from having access to resources that they should have access to. An ideal system would obviously be perfectly provisioned, with zero false positives and zero false negatives, but this is not realistically feasible since, over time, anomalies and errors are inevitable. Unlike other approaches, LAD is tunable in terms of false positives and false negatives to fit the access control use case. We discuss this tunability in Section III-E; here we motivate why this flexibility is essential, by presenting a set of potential use cases.

Detecting Anomalous Access Attempts. Certain organizations (such as the military) have strict policies that require a rigid, identity-based, mandatory access control (MAC) system despite the expense (in time and resources) of provisioning each user with specific access to a *resource*. These systems are designed such that any false positives are strictly related to errors in the manual provisioning process. In this case, our framework could be used in parallel to an existing MAC system to detect access control anomalies, e.g., instances when a user used access that he or she was provisioned, but something else about the context of that resource usage is out of the ordinary. This can be useful in scenarios such as detecting insider threats, where users who have been provisioned access behave in atypical ways prior to requesting access. Our framework generates rules based on typical access and raises flags in real time whenever these rules are violated.

Alternatively, at similar organizations where strict policies are needed, our framework could be used as a second line of defense. For example, each access control policy could be joined with the attributes derived from a secure credential (such as a Personal Identity Verification (PIV) card) reflecting the manual provisioning process described above. Our framework could prevent anomalous access in the case of errors in manual provisioning. For example, if the secure credential were stolen or forged and used without the presence of other users, without walking through the front door, or without the bearer having captured any number of *attributes* that typically are captured, our framework would be able to detect the improper access.

Learning Access Control Rules. A system administrator in charge of access to a sensitive room within a private company building may want to prevent invalid access requests and needs an appropriate access control policy to enforce this desire. Since the room is sensitive, the administrator will want to keep false positives to a minimum (potentially even at the expense of some false negatives). No probabilistic system has a complete guarantee that there will be no false positives, but we contend that less rigid organizations would trust a framework with minimal false positives over a potentially error prone manual provisioning process.

This provisioning process becomes even more complex as the number of *resources* and users grows. In fact, the same system administrator may have different needs depending on the *resource* in question. For example, access to a particular room/lab may not carry any direct security implications (e.g., any employee might have access rights to the lab). However, expected usage of this space may involve the supervision of a lab member during a particular time of day. The system administrator would prefer that access is only granted under

these conditions, but there are no security consequences otherwise. In this case, the administrator likely wants a policy with minimal false negatives (so as not to be bothered with manual access requests), even at the expense of some false positives.

Enabling Convenience. Finally, a building manager may simply want to provide convenience for occupants of a building by leveraging smart capabilities. Rather than provisioning access cards or handing out lock codes, the building manager may want users to automatically have access to appropriate elevators, rooms, or hallways. Users would have applications on their mobile devices that leverage the LAD framework in order to provide the automated access. No unintended access could occur that could not otherwise be provided by sharing a lock code or access card. In this case our framework is not for improving access control accuracy, but for providing convenience through automation.

D. Deriving Attributes and Attribute Partitions

On the user device, the beacons in the beacon repository are abstracted into an attribute set. For each access request from a user device to a specific *resource*, LAD creates of a *partition* (i.e., a subset of *attributes* relevant to the access control decision). We first describe how we capture *spatiotemporal attributes* using the beacons received from the static and mobile *nodes* and consequently, how we generate *partitions*.

In our framework, each *node* periodically transmits a discrete wireless signal with in a discrete beacon. While we deal chiefly with BLE beacons, our approach is generalizable to any discrete, periodic wireless signal. To capture an *attribute*, the user device first receives a beacon, timestamps it, and stores it in the device's local beacon repository (as shown in Fig. 1). As new beacons are received, the user device applies a set of local constraints that determine whether the received set of beacons allows the device to derive a higher level attribute. In the simplest form, a constraint may simply be "if a beacon is received from a node, an attribute with that node's identifier is added to the attribute set." More complex constraints are also possible; for instance a different constraint could be "if k consecutive beacons have been received from a single node, an attribute with that node's identifier is added to the attribute set." Even given the simple "presence" indication that beacon capture gives us, there is a wide diversity of attributes that could be specified in this way, and different user devices and resources may have different desired attributes. While LAD supports this diversity, for simplicity throughout the remainder of the paper, we assume the first attribute capture example, i.e., a single beacon received from a node results in capture of the corresponding attribute.

For practical reasons, LAD only captures positive *attributes* (i.e., a captured attribute indicates that a beacon was received). This is because there is no practical way to validate a negative *attribute* with the *authority*. A rule based on a negative *attribute* would require that the attribute is not captured which is simple to spoof by removing beacons from the beacon repository.

We have discussed how *attributes* are captured and stored at the user device. For the user device to use these *attributes* to request access to a *resource*, we need a mechanism to select the relevant *attributes*. As mentioned before, each *resource* may be interested in a separate subset of the set of *attributes* at the user device. To request access to a *resource*, a user device begins a handshake. The *resource* provides the device with a filter containing two constraints: (1) the time window over which *attributes* must be captured; and (2) a template for the *attributes* of interest. The user device constructs a *partition* of its attributes using this resource-provided filter in conjunction with any additional filters the user device has on releasing attributes (e.g., for privacy reasons). The user device then sends this *partition* to the *resource*.

The *resource* uses the partition to either grant or deny access to the user device. This decision is made via an *enforcement process* that relies on the *access control policy* at the *resource*. The details of the enforcement process are described in Section III-F. Relative to other approaches that use sensors or cameras to track users, LAD offers a privacy benefit to the user device, by allowing the device to determine what *attributes* it wants to share, and to only share *attributes* directly with the *resource* at the time of access.

We next turn our attention to the dashed lines in Fig. 1. When a user device attempts to access a *resource*, the *resource* adds the user device's provided *partition* to the resource's own log. These *partitions* are sent to the *mining service* shown in Fig 1 on a periodic basis to allow LAD's rule mining algorithm to learn the *resource*'s access control policy based on access logs. The rule mining algorithm and its parameters are described in the next section.

E. Rule Mining

In this section we describe how our framework uses the *partitions* collected at each *resource* as a training data set over which to learn access control policies. We discuss the details of our rule mining algorithm and the intuition behind our general approach to learning access control policies.

We can view the access control problem for a given *resource* to be a binary classification problem. Either the user device should be granted or denied access to the *resource*. Our first observation is that the frequency that subsets of *attributes* occur together can help us probabilistically determine a set of rules representative of the *partitions* used to gain access.

Traditional clustering algorithms, such as K-means, group feature vectors (in our case *partitions*) that are similar, while association rule mining (ARM) algorithms (such as a-priori or FP-growth), group feature dimensions (in our case the *attributes* themselves). As far as generating rules, we are primarily concerned with the latter. Specifically, we are interested in what *attributes* often occur together to allow access to a *resource*.

Therefore, our rule mining algorithm employs an ARM algorithm. For a given data set, ARM algorithms generate frequent item sets. In our case, the data set is a set of *partitions*, and each item corresponds to an *attribute*. For

example, the ARM algorithm may tell us that the item set [location_7, location_9] occurs in 20% of *partitions*, while the more restrictive item set [location_7, location_9, location_1] occurs only in 12% of *partitions*.

Our algorithm described in Algorithm 2 leverages the FP-Growth ARM algorithm to generate a list of item sets. Recall that our rules are conjunctions of *attributes*, and so each item set is a potential rule. The primary challenge of our algorithm is to reduce this list of item sets to a set of rules, combined together via disjunction, that is representative of a desired access control policy for a specific *resource*. In an ideal world, the policy would have neither false positives nor false negatives. However, leveraging former access logs to predict future access decisions is inherently probabilistic in nature. Our discussion of potential use cases demonstrates that depending on the *resource* and/or other system requirements it is useful to generate different rule sets.

To accomplish this, our rule mining algorithm uses a number of parameters, tunable via corresponding hyperparameters, to compute a utility function for each potential rule in the list of item sets. We now describe the parameters used by the utility function shown in Definition 1, and the intuition associated with each parameter.

Definition 1. Utility Function

$$utility(len, sup_{overall}, sup_{uncovered}, simil) = len^{\alpha} * sup_{overall}^{\beta} * sup_{uncovered}^{\gamma} * (1 - simil)^{\delta}$$

The parameter *len* refers to the number of *attributes* in the item set. Longer rules are more restrictive and in general lower the number of false positives, at the expense of false negatives. The parameters $sup_{overall}$ and $sup_{uncovered}$ are fractional values from 0 to 1 that represent the "support" of the item set as determined via the ARM algorithm. Referring to our previous example, if an item set appears in 12% of all *partitions*, it has a $sup_{overall}$ of 0.12. Our rule mining algorithm operates iteratively by continually selecting the next "best" item set, according to the utility function in Definition 1, to add to the rule set. This means that gradually the list of *partitions* "covered" by our rule set increases while the "uncovered" portion decreases. To represent the "support" of the item set amongst the "uncovered" attribute sets, we define the $sup_{uncovered}$ parameter. Heavier weighting of these two "support" parameters generally favors less restrictive rules. We execute our rule mining algorithm over ranges of the hyperparameters α, β, γ and δ, and compute the TPR and FPR for each permutation of parameters. Each *resource* provides the *mining service* a roc_{weight} parameter, valued from 0 to 1, to indicate the preference between TPR and FPR, for the *mining service* to select the "ideal" rule set for a given use case.

The *simil* parameter refers to the similarity of the current item set to its most similar rule currently in the rule set. Refer to Algorithm 1 for details of how this similarity metric is computed. We use a modified version of Jaccard similarity, where the importance of *attribute* intersection is weighted via the $f_{intersect}$ parameter. The $f_{intersect}$ parameter is used as another hyper parameter over which we train our rule mining algorithm.

Algorithm 1: Modified Jaccard Similarity:

```
Function SIMILARITY: itemset, f_intersect
    s = 0 for rule in ruleset do
        |I| = len(rule ∩ itemset)
        |U| = len(rule ∪ itemset)
        δ = |U| − |I|
        f = f_intersect * |I|
        s = max(s, f/(δ + f))
    end
end
```

Our rule mining algorithm is an iterative algorithm that has three terminating conditions: first, the entire list of *partitions* is covered; second, there are no more valid item sets available; and third, the utility function for the "best" rule is 0. We discuss how each of these terminating conditions is reached, and how it influences our final rule set.

1) Coverage of Attribute Sets: On each iteration of our algorithm, we recompute the utility function for each remaining valid item set, and add the item set with the highest utility score to our rule set. Depending on the hyperparameters, each rule may or may not add increase the coverage of our list of *attributes*. When the β and γ hyperparameters are higher, we generally see smaller rule sets since the *partitions* are covered more quickly.

2) Exhaustion of Item Sets: Our rule mining algorithm filters out item sets that are subsets of rules in our rule set. If the α hyperparameter is higher, we expect to see longer rules, and consequently more rules, since longer rules are typically more restrictive in terms of coverage.

3) Zero Utility Score: A zero utility score occurs when either the $sup_{overall}$ or $sup_{uncovered}$ parameters are zero. This indicates that none of the remaining valid item sets contribute to the coverage of the *partitions*.

4) Convergence: Our algorithm is guaranteed to converge since in the worst case every single item set is chosen as a rule. In practice, however, this is incredibly unlikely since one of the underlying assumptions of our framework is that we expect to see *partitions* that repeat frequently. Our evaluation validates this assumption against a real-life data set.

5) Other Parameters: Our rule mining algorithm uses one other hyperparameter not yet discussed: *min_support*. This hyperparameter is used directly by the ARM algorithm used to compute frequent item sets. It indicates the minimum "support" required for an item set to be returned for use by our rule mining algorithm. For example, if *min_support* is 0.1, then only item sets that "cover" at least 10% of the attribute sets are considered as potential rules.

F. Enforcing Access Control Rules

Rule enforcement is a matter of collecting the relevant *partition* from a user device at the time an access request is made. Fig. 1 shows the components involved in validating

Algorithm 2: Rule Mining Algorithm:

Input: $\alpha, \beta, \gamma, \delta, f_{intersect}, min\_support$
Output: $ruleset$
$\vec{i}, \overrightarrow{len}, \overrightarrow{sup_{overall}}$ = **FP-Growth** $(\overrightarrow{partition}, min\_support)$
uncovered = $\overrightarrow{partition}$
$ruleset = \emptyset$
while $|uncovered| > 0$ *and* $|\vec{i}| > 0$ **do**
  $\vec{i}, \overrightarrow{len}, \overrightarrow{sup_{overall}}$ = **remove_subsets**$(\vec{i}, ruleset)$
  $\overrightarrow{simil}$ = **similarity**$(\vec{i}, f_{intersect})$
  $\overrightarrow{sup_{uncovered}}$ = **compute_uncovered**$(\vec{i},\ uncovered)$
  $\vec{u}$ =
   utility$(\vec{i}, \overrightarrow{len}, \overrightarrow{sup_{overall}}, \overrightarrow{sup_{uncovered}}, \overrightarrow{simil}, \alpha, \beta, \gamma, \delta)$
  $ruleset \leftarrow \max(\vec{u})$
  $uncovered$ = **prune**$(uncovered, ruleset)$
end
return $ruleset$

the *partition*. The *partition* is first sent from the user device to the *resource*. The *resource* sends the *partition* to the *authority* to validate the fidelity of the *attributes*. Finally, the *resource* determines whether the *partition* satisfies the policy and sends the access decision back to the user device.

Our attributes are derived from BLE beacons that like all wireless signals are susceptible to overhearing. In order to prevent attribute spoofing, our framework requires that BLE beacons change their identifiers over time. The *authority* maintains a temporal mapping such that beacons retrieved outside the specified time window are invalidated.

G. Threat Model

Our framework is designed to prevent unauthorized access to *resources* that are protected via access control policies. From an authentication perspective, our approach does no worse than traditional approaches to authentication that rely on provisioned credentials such as personal identity verification (PIV) cards and RSA SecurID software and hardware tokens. All beacons from the environment used in the capturing of *attributes* are verified using these credentials. In fact, our framework inadvertently improves authentication since our *attributes* serve as additional factors for authentication.

We assume that users are trusted in so far as any members of an organization are trusted. For example, employees at a company or faculty/staff at a university are generally trusted to have little incentive to participate in collusion or replay attacks. Our system is no more susceptible to attacks, than a scenario in which an employee decides to share a PIV card or SecurID token with an outside party or another employee.

From the perspective of a single party, as discussed in Section III-F, we prevent temporal forging of context by using rolling / time-based ephemeral identifiers. Specifically, the BLE beacon identifiers change value over time. In this way, access is revoked over time, since beacons retrieved during a given time interval cannot be used to derive *attributes* to provide access at a different time interval.

IV. Evaluation

In this section, we describe how we evaluated the LAD framework using a real-life, testbed deployed throughout a multi-story academic building. First, we discuss the details of our deployment, and the associated experimental setup. We then evaluate the accuracy of our rule mining algorithm against ML approaches for binary classification. We compare against decision tree based approaches, XGB and RF, that are the state of the art for small to medium sized datasets like those at the "building-scale," the target of the LAD framework.

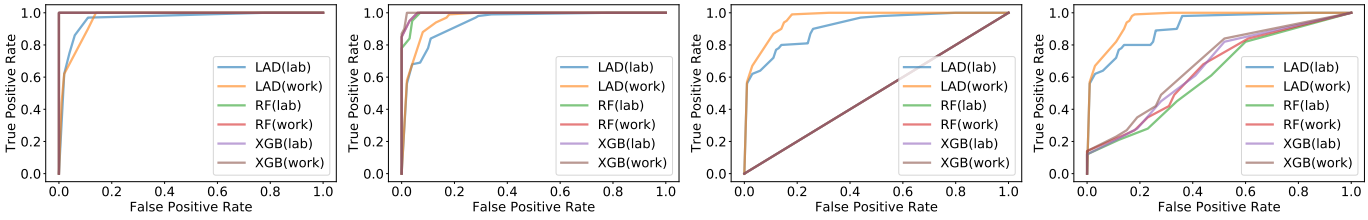
A. Deployment and Experimental Setup

We deployed an extensive sensor testbed across 4 floors of an engineering building on our campus. We deployed 24 static Nordic Thingy devices [1] across the building and recruited 7 mobile participants to carry the same devices over the course of 45 days. The Nordic Thingy is an inexpensive, multi-function low-power sensor built around the nRF52832 Bluetooth 5 SoC and capable of transmitting BLE beacons. We flashed these devices with a continuous neighbor discovery protocol to make them periodically transmit unique BLE beacons. The 7 mobile participants were also asked to carry Moto E 2nd generation Android devices running Android 5.1. These devices scanned continuously in the background for BLE beacons from the Nordic Thingy devices. Within our framework, these Nordic thingy devices serve as *nodes* that transmit beacons to be received by the Android devices and processed into *attributes*. The static *nodes* are placed near *resources* and common areas where we expected human foot traffic. Unlike indoor localization systems, our deployment does not necessitate arduous effort in determining *node* placement. The LAD framework functions within an inexpensive smart environment both in terms of *node* cost and deployment labor. We collected over 10 million BLE beacons over the course of the experiment (the dataset is publically available [32]). The richness of the dataset is a secondary contribution of our work that we aim to leverage in the future.

We also placed 3 Android Nexus 9 tablet devices in locations of common use and interest to our participants. Our participants were instructed to "check-in" at these devices by pressing their names when they entered the area. These tablets serve as *resources* to which the participant devices request access. Each check-in generates a *partition* at a *resource* based on the *attributes* collected by the participant device prior to the time of check-in. Tablet A corresponds to a work area where the 7 participants spend some portion of their day working and hold impromptu meetings. Tablet B and Tablet C correspond to opposite doors of a lab area where the 7 participants spend a smaller portion of their day, and hold scheduled meetings. Since Tablet B and Tablet C logically restrict access to the same *resource*, we grouped these check-ins together.

From here on, we refer to the *resource* at Tablet A as *Work*, and the *resource* at Tablets B and C as *Lab*. Across the 7 participants, we observed 642 check-ins at *Work*, and 317

[1] https://www.nordicsemi.com



(a) Absent **(b)** Known **(c)** Unknown **(d)** Injected Anomalies

Fig. 2: Receiver operating characteristic (ROC) curves for LAD, XGB and RF for providing access to Lab and Work resources for varying types of access control violations

check-ins at *Lab*. To evaluate LAD's performance we split our data set into two sets, consecutive in time. We wanted to measure that the rules generated during the first part of the experiment were applicable to participant behavior in the latter part. We used the first 21 days of the experiment for the training set, gathering *partitions* at the *resources* from participant devices. Our test set comprises the latter 24 days of the experiment, where instead we use the *partitions* from participant devices to determine whether access should be granted or denied.

In the subsequent sections we demonstrate LAD's performance against varying slices of this dataset, and against existing ML approaches. Recall that existing work in mining rules for ABAC [15], [21] is not suitable to *attributes* derived from contextual data that is constantly changing and inherently noisy. These types of *attributes* necessitate a probabilistic approach to rule learning, which is why we compare against ML approaches that offer the closest alternative.

To simplify discussion, we define two possible labels for each *partition* of *attributes* in our dataset: (1) valid implies that the *partition* should grant access, and (2) invalid implies that *partition* should not grant access.

B. Performance in Absence of Access Control Violations

By nature of typical user interaction with *resources*, we expect the majority of access requests to be valid. Therefore, for our first experiment, we label each *partition* at the time of checkin as valid. LAD, XGB and RF all employ parameterizable learning models. As such, to compare performance, we generate receiver operating characteristic (ROC) curves for all 3 approaches against both the *Work* and *Lab* resources.

Since there are no invalid labels in the training set, we expect XGB and RF to mark each checkin in the test set as valid. This is reflected in Fig. 2a where XGB and RF achieve perfect performance for both resources (these 4 curves are overlapping). This experiment serves as a good benchmark for LAD since the rules generated from the training set will not be completely representative of behavior during testing. While LAD does not exhibit perfect performance, it is not far behind. For example, it achieves a 90% TPR at an 11% FPR for *Work*. The experiment shows that LAD is functional and useful in the "steady-state" where there are no invalid access attempts.

C. Performance against Known Types of Access Control Violations

While LAD does not require invalid labels to learn its access control policy, we wanted to demonstrate that even in the presence of an external mechanism for specifying *partitions* that are labeled invalid, where we expect XGB and RF to shine, LAD performs comparably. For example, in the future, labeled data including invalid labels may available from a similar environment, and adapted to the current environment.

To design this experiment, we manually labeled all *partitions* that involved entry into the building as valid, and labeled all other partitions *invalid*. This is deliberately a perfect scenario for decision tree based ML approaches. Meanwhile, LAD is not perfectly suited because rules are generated based on *nodes* along the user's entire path through the building, even when the *attribute* associated with a *node* is not critical to labeling a *partition* as valid or invalid.

As shown via the ROC curve in Fig. 2b, and as expected, XGB and RF outperform LAD. For example, for *Work*, XGB achieves a 90% TPR with only a 1% FPR while LAD achieves a 90% TPR with an 8% FPR. This experiment demonstrates some of the flexibility of LAD since it performs reasonably well even in a scenario for which it is not perfectly suited.

D. Performance against Unknown Types of Access Control Violations

Invalid access attempts occur infrequently. Moreover, access control systems desire to detect these invalid access attempts before they ever occur. We construct two experiments that demonstrate LAD's performance in face of two related scenarios: (1) no invalid *partitions* are used to request access during training, but invalid *partitions* are used to request access during testing, and (2) some invalid *partitions* are used to request access during training, but new, unpredictable invalid *partitions* are used to request access during testing.

For the first experiment, we modify the experiment in section IV-C by removing all of the invalid *partitions* from the training set. The goal is to see if we distinguish between valid and invalid *partitions* in the training set, without apriori knowledge regarding what may constitute an invalid *partition*. Without invalid access attempts to learn from, XGB and RF fail to classify future invalid access attempts (as shown by the 4 overlapping diagonal lines in Fig. 2c). Meanwhile, LAD exhibits substantially superior performance, and performs similarly to how it did in the previous experiments. For example,

LAD achieves a 99% TPR with a 18% FPR for *Work* in both Fig. 2b and Fig. 2c. Note that for use cases more sensitive to FPR, we still achieve a 67% TPR with a 3% FPR. The experiment demonstrates LAD's power in detecting invalid access attempts without ever having seen any.

For the second experiment, we modify the experiment in section IV-C by injecting additional invalid *partitions* into the testing set. These *partitions* are generated by walking atypical paths from outside the building to *Work* and *Lab*. These paths emulate an intruder who might enter the building through the back door and proceed up the stairs instead of using the elevator to remain unseen. The goal is to see how performance changes if there exist some invalid *partitions* on which to learn.

Fig 2d shows that XGB and RF improve marginally relative to the prior experiment. This is because the training set consists of *partitions* labeled both valid (involving building entry) and invalid (not involving building entry) instead of only those labeled valid. However, yet again, XGB and RF are unable to perform as well as LAD in the face of "new" invalid access attempts such as those from the atypical paths. For example, for *Work*, XGB achieves only a 23% TPR with a 10% FPR while LAD achieves a 82% TPR with a 10% FPR. This experiment further demonstrates LAD's power in detecting unpredictable anomalies.

V. Conclusions and Future Work

In this paper, we showed how LAD is able to automatically generate spatiotemporal attributes for use in learning and enforcing access control policies. We demonstrate that a probabilistic approach to mining policies can be effective and flexible, making it applicable to a wide range of use cases. We evaluated our approach by deploying a real-life, building-scale testbed including active human users and passive static sensors. In the future, we hope to expand the attribute space of our approach by deriving meta-attributes from our existing ones, and by evaluating LAD against a wider range of *resources*. In conclusion, LAD provides an important step in the direction of building flexible, automated access control systems that are applicable at the building-scale.

Acknowledgements

This work was funded in part by the National Science Foundation, Grant #CNS-1703497.

References

[1] M. Miettinen *et al.*, "Conxsense: automated context classification for context-aware access control," in *Proc. of the 9th ACM symposium on Information, computer and communications security*, ACM, 2014.

[2] E. Hayashi, S. Das, S. Amini, J. Hong, and I. Oakley, "Casa: Context-aware scalable authentication," in *Proc. of ACM SOUPS*, ACM, 2013.

[3] K. Olejnik *et al.*, "Smarper: Context-aware and automatic runtime-permissions for mobile devices," in *IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017.

[4] F. Paci, A. Squicciarini, and N. Zannone, "Survey on access control for community-centered collaborative systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, p. 6, 2018.

[5] S. Osborn, "Mandatory access control and role-based access control revisited," in *Proc. of ACM Workshop on Role-based Access Control*, Citeseer, 1997.

[6] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, and J. Voas, "Attribute-based access control," *Computer*, vol. 48, no. 2, pp. 85–88, 2015.

[7] M. L. Mazurek *et al.*, "Toward strong, usable access control for shared distributed data," in *Proc. of the 12th USENIX conf. on File and Storage Technologies*, pp. 89–103, USENIX, 2014.

[8] C. Julien, C. Liu, A. L. Murphy, and G. P. Picco, "Blend: practical continuous neighbor discovery for bluetooth low energy," in *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 105–116, ACM, 2017.

[9] Y. Qiu, S. Li, X. Xu, and Z. Li, "Talk more listen less: Energy efficient neighbor discovery in wireless sensor networks," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, IEEE, 2016.

[10] L. Wei, B. Zhou, X. Ma, D. Chen, J. Zhang, J. Peng, Q. Luo, L. Sun, D. Li, and L. Chen, "Lightning: A high-efficient neighbor discovery protocol for low duty cycle wsns," *IEEE Communications Letters*, vol. 20, no. 5, pp. 966–969, 2016.

[11] T. Kalbarczyk and C. Julien, "Omni: An application framework for seamless device-to-device interaction in the wild," in *Proceedings of the 19th International Middleware Conference*, pp. 161–173, ACM, 2018.

[12] C. Liu, J. Hua, and C. Julien, "Scents: Collaborative sensing in proximity iot networks," in *Proc. of IEEE PerCom Workshops*, 2019.

[13] C. Liu, C. Julien, and A. Murphy, "PINCH: Self-organized context neighborhoods for smart environments," in *Proc. of IEEE SASO*, 2018.

[14] A. Montanari, S. Nawaz, C. Mascolo, and K. Sailer, "A study of bluetooth low energy performance for human proximity detection in the workplace," in *Proc. of PerCom*, 2017.

[15] Z. Xu and S. D. Stoller, "Mining attribute-based access control policies," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 533–545, 2014.

[16] R. Agrawal, R. Srikant, *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, pp. 487–499, 1994.

[17] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Acm sigmod record*, vol. 22, pp. 207–216, ACM, 1993.

[18] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM sigmod record*, vol. 29, pp. 1–12, ACM, 2000.

[19] M. L. Mazurek *et al.*, "Access control for home data sharing: Attitudes, needs and practices," in *Proc. of SIGCHI*, ACM, 2010.

[20] A. Gupta *et al.*, "Intuitive security policy configuration in mobile devices using context profiling," in *2012 International Conf. on Privacy, Security, Risk and Trust*, IEEE, 2012.

[21] Z. Xu and S. D. Stoller, "Mining attribute-based access control policies from rbac policies," in *2013 10th International Conf. on Emerging Technologies for a Smarter World (CEWIT)*, pp. 1–6, IEEE, 2013.

[22] C. Zhang, J. Han, L. Shou, J. Lu, and T. La Porta, "Splitter: Mining fine-grained sequential patterns in semantic trajectories," *Proc. of VLDB Endowment*, vol. 7, no. 9, pp. 769–780, 2014.

[23] C. Zhang *et al.*, "Gmove: Group-level mobility modeling using geo-tagged social media," in *Proc. of SIGKDD*, pp. 1305–1314, ACM, 2016.

[24] N. Zhou *et al.*, "A general multi-context embedding model for mining human trajectory data," *IEEE transactions on knowledge and data engineering*, vol. 28, no. 8, pp. 1945–1958, 2016.

[25] N. Rajagopal *et al.*, "Enhancing indoor smartphone location acquisition using floor plans," in *Proc. of IPSN*, IEEE, 2018.

[26] D. Oosterlinck, D. F. Benoit, P. Baecke, and N. Van de Weghe, "Bluetooth tracking of humans in an indoor environment: An application to shopping mall visits," *Applied geography*, vol. 78, pp. 55–65, 2017.

[27] P. Lazik *et al.*, "Alps: A bluetooth and ultrasound platform for mapping and localization," in *Proc. of ACM SenSys*, ACM, 2015.

[28] F. Maiorano and A. Petrosino, "Granular trajectory based anomaly detection for surveillance," in *Proc. of ICPR*, IEEE, 2016.

[29] J. Xiao *et al.*, "A survey on wireless indoor localization from the device perspective," *ACM Computing Surveys (CSUR)*, 2016.

[30] C. Piciarelli *et al.*, "Trajectory-based anomalous event detection," *IEEE Transactions on Circuits and Systems for video Technology*, 2008.

[31] Y. Chen, P. Yuan, M. Qiu, and D. Pi, "An indoor trajectory frequent pattern mining algorithm based on vague grid sequence," *Expert Systems with Applications*, vol. 118, pp. 614–624, 2019.

[32] C. Liu, J. Hua, T. Kalbarczyk, S. Lee, and C. Julien, "Dataset: User side acquisition of People-Centric sensing in the Internet-of-Things," in *Second workshop on Data Acquisition To Analysis*, (New York, NY), ACM, 2019.