Ancile: A Practical System for Supporting Privacy-Aware Ubiquitous Computing Applications

ANONYMOUS AUTHOR(S)

Widespread deployment of Intelligent Infrastructure and the Internet of Things enables new ubiquitous computing applications, while posing new privacy threats. *Ancile* manages use-based privacy for the sensitive data consumed by these applications. We focus on third-party, location-based, enterprise applications illustrating how Ancile can enable developers to efficiently implement, and enterprises to safely deploy, such applications.

CCS Concepts: • **Information systems** → **Location based services**; *Enterprise applications*; • **Security and privacy** → **Privacy-preserving protocols**; • **Computer systems organization** → *Sensor networks*;

Additional Key Words and Phrases: Privacy Policy, Use-Based Privacy, Ubiquitous Computing

ACM Reference Format:
Anonymous Author(s). 0. Ancile: A Practical System for Supporting Privacy-Aware Ubiquitous Computing Applications. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 0, 0, Article 0 (0), 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Available sensing technologies have fueled demand for more efficient, flexible, and digitally-enabled infrastructure. Data available from the resulting "intelligent infrastructures" open up new opportunities for ubiquitous computing applications in support of enterprise and individual objectives—collaboration and productivity to energy-efficiency and wellness. As was the case for mobile and social applications, ubiquitous computing applications are increasingly developed by third-parties—e.g., apps that use Amazon Skills and Slackbot APIs. However, deployment of third-party applications over intelligent infrastructure raises privacy concerns.

Use-based privacy [5, 6, 8, 9, 14, 27] dictates how data may be used in a given context, re-framing privacy as the prevention of harmful use rather than as passive access control based on informed consent. The approach is well-suited to pervasive data collection performed by intelligent infrastructures, but there are currently no tools available for application developers to demonstrate that sensitive data—collected by the infrastructure and consumed the application—are used in a privacy-compliant manner.

In an effort to bridge this gap, this paper investigates the following:

(1) What gives developers a practical way to write ubiquitous computing applications in a privacy-aware manner?
(2) How can we design a software platform that supports rich use-based policies for enterprises that run these applications?
(3) What is the overhead of using such a platform and what are its limitations?

To explore these questions, we developed, *Ancile* (Figure 1), a system that augments existing intelligent infrastructures with enforcement mechanisms for use-based privacy. We deploy Ancile using a campus-wide

2474-9567/0/0-ART0 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

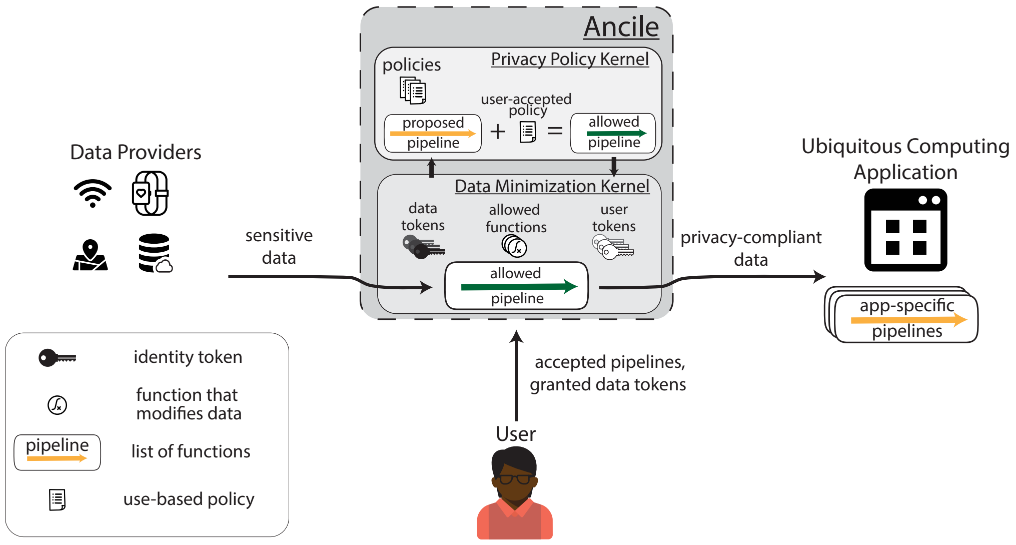


Fig. 1. Ancile is designed to enable application developers to efficiently implement, and enterprises to safely deploy, *privacy-aware* applications using fine-grained use-based privacy rules. To request users' data, applications specify pipelines—lists of functions that request, process, and filter data. Ancile maintains data tokens to data providers and can execute proposed pipelines and enforce that submitted pipelines satisfy policies accepted by the enterprise and users. Section 3 describes these components in depth.

location service—location information is the most prevalent, high utility, and high sensitivity data sources available [3, 4, 15, 22]—and we evaluate Ancile by prototyping three light-weight, ubiquitous computing applications running under a variety of simulated work loads. Our initial findings suggest that the Ancile approach is both expressive and scalable. We find Ancile overhead to be below 100 ms for each of our developed applications and that Ancile can process more than 700 requests per second.

The rest of the paper is organized as follows. We describe a trust model and our use-based privacy framework in Section 2. Section 3 describes the design of Ancile and we present an initial evaluation of our system in Section 4. We discuss related work in Section 5 and finish with a discussion our future work and conclusions in Section 6.

2 TRUST AND PRIVACY FRAMEWORK

2.1 Trust Model

Ancile runs on centrally-managed enterprise resources, comprising of data providers, applications, and end-users, which we defined below.

- *Data Providers* are services that handle sensor-derived (e.g., location) or user-generated (e.g., Gmail or Slack) data. Data providers run on centrally-managed trusted enterprise resources.
- *Applications* consume data stored by data providers to implement specific services or functions. Application requests for data are handled by Ancile to ensure privacy compliance. Applications (whether intentionally or due to programmer error) might attempt to perform actions that are incompatible with a privacy policy governing how data may be used, so use of information obtained by data providers by applications is considered untrusted. However, we do trust applications to only act on behalf of approved users (for a

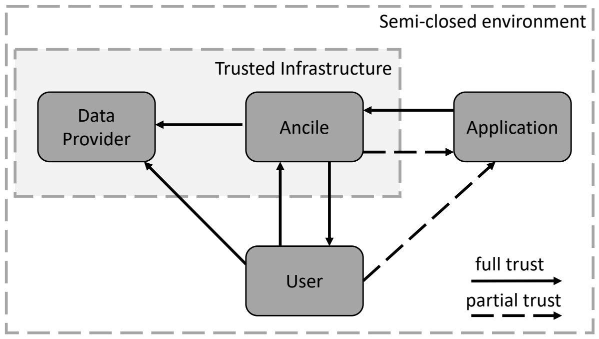


Fig. 2. Trust model. Ancile is fully trusted by users and applications to handle sensitive data. However, users do not need to trust applications with their data directly, as Ancile enforces use of the data to comply with accepted policies.

given purpose); such approvals are managed outside of Ancile—e.g., when a user installs a Slackbot on their local machine.

- *Users* are principals who request a service or run an application. Users are also the subjects of data collected, generated, and stored by data providers.

Interactions between these principals in the Ancile ecosystem are summarized in Figure 2.

2.2 Use-based Privacy

Ancile enforces use-based privacy for data collected and generated by data providers. In contrast with traditional access-based approaches—which focus on limiting data collection—use-based privacy languages [6, 9] express restrictions on how data may be used by applications. This approach aligns well with the challenges of ubiquitous computing, where data providers often collect sensitive data through pervasive physical sensors embedded in the environment.

Enforcing a meaningful notion of privacy requires imposing strict limits on how data may be used, while still supporting third-party applications. Ancile meets this challenge by expressing use-based privacy policies in the Avenance policy language[5, 6], a *reactive* language for expressing use-based privacy policies. A reactive language is one in which the current set of use restrictions depends not only on the initial data and its type, but also on the history of events that might have affected that data. For example, a policy might permit a user's location during work hours to be used permissively, but impose use-restrictions during evenings and weekends. Or policies might restrict the use of raw location data, only allowing enterprise-controlled Data Providers to perform particular filtering, authorizing the output of those functions for use by third party applications.

In the Avenance language, data are tagged with use authorizations. Current use-authorizations are expressed as triples (I, P, E), where I is an invoking principal (an application), P is a purpose (a reason to get the data), and E is an executable (an action that may be performed). The set of possible use-authorizations forms a finite state automaton; the state of this privacy automation changes when contextual events or data transformations occur, and authorization decisions are based on the current state of the privacy automata. The resulting authorization policies impose reactive use-restrictions. For example, Figure 3(a) shows a simple policy that allows *Application* to perform a filter action—e.g., remove location data that represents a location outside the building—on raw location data, and then permits the resulting filtered-location data to be viewed by *Application.* Figure 3(b) depicts a policy that allows current location data to viewed by *Application* between the hours of 9:00 and 18:00, thereby protecting the privacy of sensitive location information outside of official work hours.

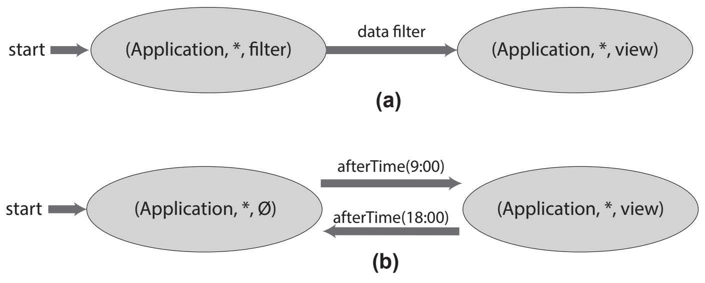


Fig. 3. Example of state automate used for Use-Based privacy. (a) simple filter policy that prevents data access without filtering; (b) a policy that allows data access only during certain time period.

3 SYSTEM DESIGN

We implemented Ancile as a run-time monitor positioned between applications and sensitive user data. The system is responsible for authorizing applications' requests for sensitive data. If a request is authorized, then Ancile performs the data look-up and executes transformation operations (e.g., filtering) on the requested data. We split our Ancile implementation, shown in Figure 4, into two components: the *Data Minimization Kernel (DMK)* and the *Privacy Policy Kernel (PPK)*. This design helps avoid contention for resources and provides flexibility for various kinds of deployments. The DMK is responsible for handling requests from applications, communicating with data providers, executing requested operations, and logging requests; the PPK makes authorization decisions.

3.1 Use-based Privacy in Ancile

Administrator-defined *mandatory policies* express use restrictions that apply to all data of a specified type. For example, the PI of a research study that places restrictions on how collected data may be used might define a policy that specifies those restrictions. These restrictions might require data to be processed or filtered in a particular way (e.g., Figure 6), or they might specify contextual limitations (e.g., time-bounds) on when data may be accessed (e.g., Figure 7). Mandatory policies impose a baseline set of restrictions about how data stored by a data provider may be used.

Users also grant use authorizations in response to application requests. To issue such a request, an application constructs a *pipeline*—a sequence of data transformations that the application would like to have performed. Algorithm 1 (below) shows an example pipeline—the application filters location data by replacing fine-grained coordinates (e.g., latitude and longitude) with a floor number. If a user accepts the pipeline request, then Ancile generates the Avenance policy that expresses the new authorization and adds the new policy to the user's data. Figure 6 shows an example of such policy generated from a pipeline [*get_location*, *filter_data*, *return_data*]. Ancile's Pipelines, which were not present in previous implementations of Avenance [5, 6], make it possible to incorporate application-specific data processing without having to sacrifice the guarantees captured in use-based privacy policies.

```
ALGORITHM 1: Pipeline example
1. get_location – get data from campus location service;
2. filter_data – remove $(x, y)$ coordinates from the received data, leave just floor data;
3. return_data – return data to an application;
```

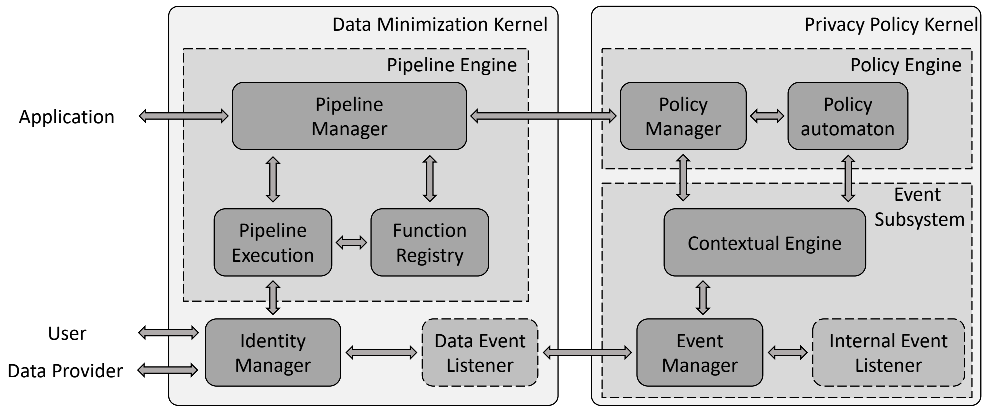


Fig. 4. Overview of the Ancile system. The *Data Minimization Kernel* is processes and executes an application's pipeline requests, and manages users' identity. The *Privacy Policy Kernel* checks that pipelines comply with associated policies and controls updates of policies with contextual events.

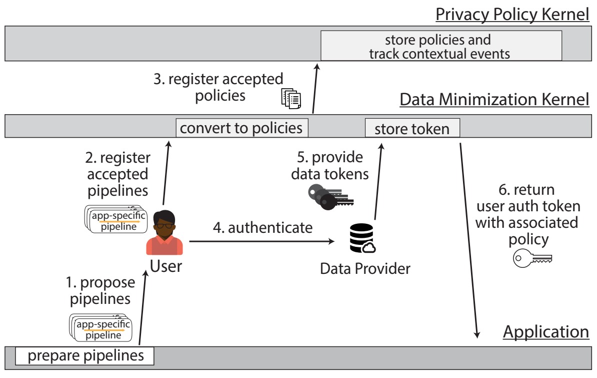
Pipeline execution is authorized by Ancile only if the user has approved the pipeline and that pipeline is compliant with all applicable mandatory policies.

3.2 Data Minimization Kernel

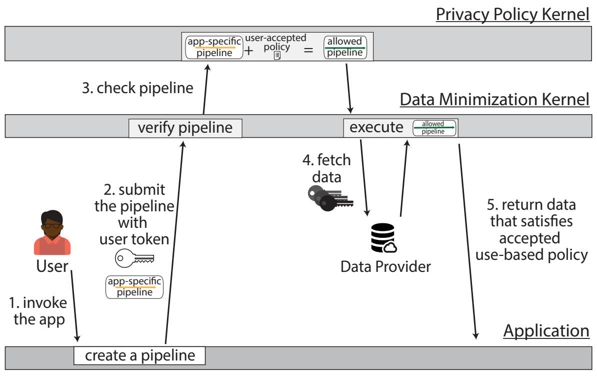
The DMK handles identity management and data transformation tasks for an application. It uses the *Pipeline Engine* to track and execute permitted pipelines and allowed transformations. However, verification of pipelines with respect to policies is handled by the PPK. The DMK uses a *Function Registry* to store data transformation operations that are available to developers and can be used to specify a pipeline. Concretely, each function comprises a block of custom code that must be manually reviewed before being added to the Function Registry. We believe this overhead is acceptable, as the Ancile deployment model assumes supervision of enterprise's IT department and, therefore, can be supported by local IT staff.

When a pipeline requests data from a Data Provider, the DMK uses the *Identity Manager* to store data authorization tokens that users provide to Ancile. To ensure communication security, Ancile uses OAuth2 to manage access tokens for data providers and network connections are secured by SSL. Ancile is itself an OAuth2 provider to apps and issues user tokens to the application when users accept proposed pipeline. The issued token is associated with the accepted policy; when the application requests data using this token, it pulls the associated policy.

Figure 5 illustrates two modes of interaction between an application and Ancile: (a) pipeline registration and (b) pipeline execution. In the first mode, registration of user-accepted pipelines invokes the DMK, which converts these pipelines into use-based policies, gathers required data authorization tokens, and returns a new user token to the application. This user authorization token allows an application to request execution of a pipeline by Ancile for the given user. To simplify efforts to integrate with Ancile, we don't consider adding policies for contextual events by applications; as these policies are defined by enterprises and can be added later to the existing user's policy. In the second mode, pipeline execution, the application reacts to user's request, for example booking a



(a) Registration of the app and pipelines for a user;



(b) Exucution of a pipeline on user's call

Fig. 5. Executing an Application Pipeline. (a) The application has only to propose list of pipelines to a user and Ancile will manage identity management and handle a single token for every user (b) The application creates a new pipeline according to its logic and Ancile executes this pipeline if it complies with user-accepted policy and returns the resulting data to the application.

Table 1. Privacy Policy Kernel exposed API commands that are available to Data Minimization Kernel

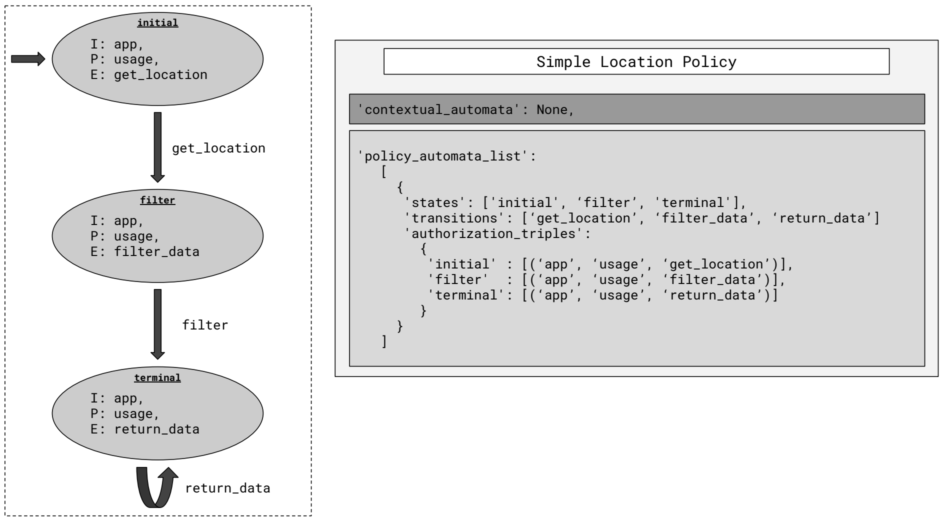
API Command	Arguments	Description
Create	(policy)	Enter the given policy into the database and return ID
Delete	(ID)	Remove the policy associated with the given ID
Authorize	(proposed actions, ID)	Evaluate the proposed actions on the policy

room, and submits a pipeline through the user token which is executed after the policy is checked by the Policy Kernel.

3.3 Privacy Policy Kernel

The PPK is responsible for managing, evaluating, and updating the privacy policies attached to a user's data in the system. It is a run time monitor, and it consists of a *Policy Engine*, which validates policy requests, and an *Event Subsystem*, which handles contextual events of dynamic policies. The request API contains three instructions: Create, Delete, and Authorize, which are summarized in Table 1. The first two are administrative instructions, while the third instruction, authorize, is the primary functionality exposed by the PPK. The Authorize instruction is the means by which the DMK can ask if a proposed pipeline of actions is compliant with a given policy. When a request made to Ancile, this instruction will be called to ensure compliance with all relevant policies attached to a given piece of data.

The PPK uses the *Policy Automaton* to track the state of each policy in conjunction with the *Event Subsystem*. The *Contextual Engine* handles event updates which are sent from and configured by the *Event Manager*. It sets asynchronous event triggers in *Internal Event Listeners* which listen for events like time changes that can be used to limit application data access attempts. Additionally, *Data Event Listeners* in the DMK are triggered when a policy depends on changes, such as calendar updates or location changes.



```
Simple Location Policy

'contextual_automata': None,

'policy_automata_list':
   [
     {
      'states': ['initial', ‘filter’, 'terminal'],
      'transitions': [‘get_location’, ‘filter_data’, ‘return_data’]
      'authorization_triples':
        {
         'initial' : [(‘app’, ‘usage’, ‘get_location’)],
         'filter'  : [(‘app’, ‘usage’, ‘filter_data’)],
         'terminal': [(‘app’, ‘usage’, ‘return_data’)]
        }
     }
   ]
```

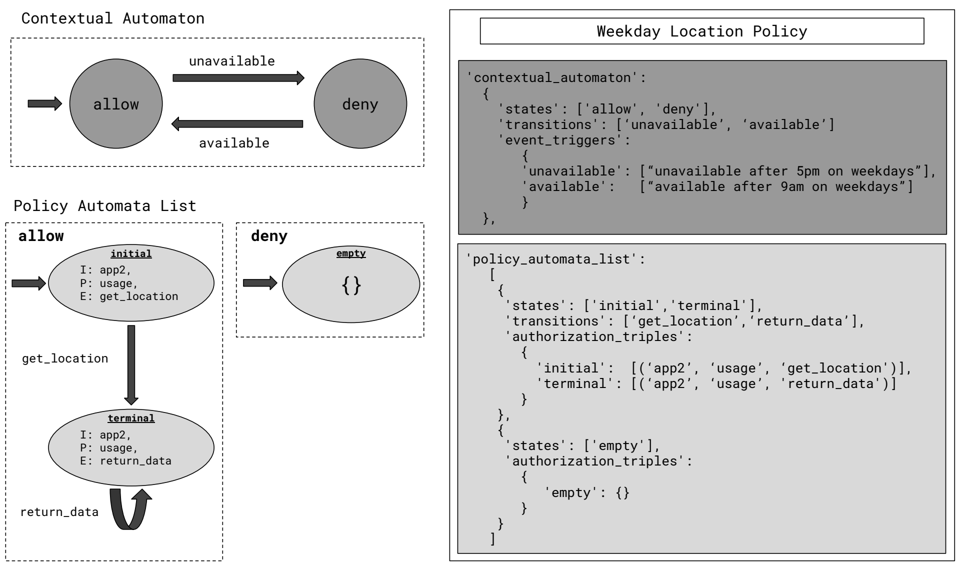
Fig. 6. An example of a simple static policy rendered both in its privacy automaton and data structure forms. The policy allows the pipeline [*get_location, filter_data, return_data*] and denies anything else. The final "operation", *return_data* is not an operation the DMK executes, but an indicator that the transformed data at this point is allowed to leave Ancile.

3.4 Tradeoffs between data minimization and application-implementation overhead

The Ancile DMK enables applications to delegate certain data processing steps to Ancile, which is trusted to execute each pipline in a privacy-preserving manner. This design raises an interesting question: how should application logic be partitioned between the DMK and the application itself? To illustrate, consider a location-aware room booking application that first retrieves the user's location and then books a nearby room from a third-party calendar application. One approach would be to perform both operations—location retrieval and room booking—using a single Ancile pipeline. In this case, the application is never sent the user's location, only the room that was successfully booked. A different approach is to simply filter the location data within Ancile, and return it to the application, which then performs the calendar booking. More specifically, the DMK pipline might retrieve the user's fine-grained location and map it to a coarse-grained location (e.g., a building floor). Then, once the application decides which room to book on that floor, it might use another pipeline to book the room. The first approach satisfies a stronger set of use-based privacy policies, but requires the developer to place their entire application logic within Ancile. In practice, we believe this would require excessive support to develop custom pipelines for each application. In contrast, the second approach implements more of the application logic within the application itself, but satisfies a weaker set of use-based policies, since the building floor is provided directly to the application.

4 EVALUATION

In this section, we explore Ancile's functionality and performance by testing a prototype in support of three location-based applications. We evaluate the overhead introduced when running these applications and load-test individual components (PPK and DMK) to evaluate how Ancile operations scale in an enterprise setting.



```
'contextual_automaton':
  {
    'states': ['allow', 'deny'],
    'transitions': ['unavailable', 'available']
    'event_triggers':
       {
       'unavailable': ["unavailable after 5pm on weekdays"],
       'available':   ["available after 9am on weekdays"]
       }
  },
```

```
'policy_automata_list':
   [
    {
     'states': ['initial','terminal'],
     'transitions': ['get_location','return_data'],
     'authorization_triples':
       {
         'initial':  [('app2', 'usage', 'get_location')],
         'terminal': [('app2', 'usage', 'return_data')]
       }
    },
    {
     'states': ['empty'],
     'authorization_triples':
       {
          'empty': {}
       }
    }
   ]
```

Fig. 7. An example of a simple policy with a contextual component rendered in both forms. Three state machines are used; one captures contextual changes while the other two represent the policy to be evaluated at that contextual state.

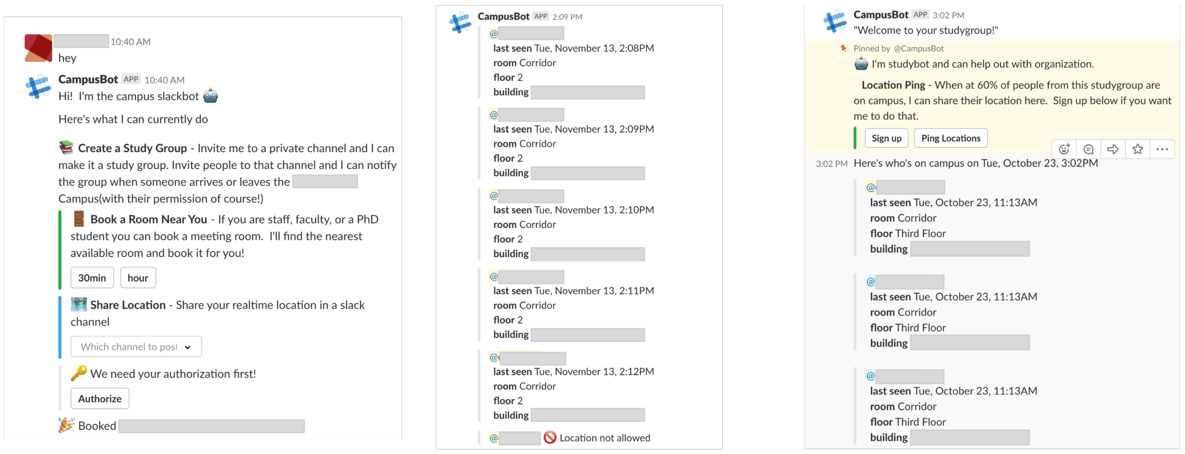
4.1 Experimental setup

Sample Applications Setup: We run our Ancile applications on a single Amazon Web Services (AWS) t2.micro instance [37], which runs with a single vCPU and 1 GB of RAM. Our applications are built using the Elixir language and Phoenix web framework, and they are implemented as a Slackbot [36] that can be deployed in the enterprise's Slack workspace by the Slack administrator. To estimate a user's indoor position, we use location services provided by campus-wide deployed Aruba 7240 Wireless Controllers with ArubaOS version 6.5.3.7 and Aruba Access Points AP315 [34]. Location services provide data to Ancile through a web service using OAuth tokens to manage access to the data. To book rooms using Ancile, we use campus Microsoft Exchange server and Graph API, which we treat as a data provider.

Load Testing Setup: We evaluate Ancile under various request loads when the DMK and the PPK are deployed on separate machines. The DMK runs on a VM with 4 cores on an Intel(R) Xeon(R) E5-2620 v2 @ 2.1GHz with 8 GB of RAM, while the PPK runs on a test machine with an Intel i7-4790 @ 3.6 GHz Quad Core processor and 8 GB of RAM. We choose these machines as a typical enterprise hardware available, and we place them on the same local network. To generate a test load for the PPK, we simulate 400 clients across 50 local machines using a custom Python script and have them make simultaneous requests. New requests are generated at a fixed intervals, and each client retains timing information for every sent request. To test the DMK, we used the wrk benchmark tool [41] and a custom web request.

4.2 Location-Aware Slackbot applications

We focus on location-aware ubiquitous computing applications such as those that leverage WiFi access points to estimate user's precise position within a building [42]. We implement the following location-based applications in the form of Slackbots to illustrate the functionality and test the performance of Ancile.


[redacted] 10:40 AM
hey
CampusBot APP 10:40 AM
Hi! I'm the campus slackbot
Here's what I can currently do
Create a Study Group - Invite me to a private channel and I can make it a study group. Invite people to that channel and I can notify the group when someone arrives or leaves the [redacted] Campus(with their permission of course!)
Book a Room Near You - If you are staff, faculty, or a PhD student you can book a meeting room. I'll find the nearest available room and book it for you!

Share Location - Share your realtime location in a slack channel

We need your authorization first!

Booked [redacted]

(a) Main menu and BookNearMe

CampusBot APP 2:09 PM
@[redacted]
last seen Tue, November 13, 2:08PM
room Corridor
floor 2
building [redacted]
@[redacted]
last seen Tue, November 13, 2:09PM
room Corridor
floor 2
building [redacted]
@[redacted]
last seen Tue, November 13, 2:10PM
room Corridor
floor 2
building [redacted]
@[redacted]
last seen Tue, November 13, 2:11PM
room Corridor
floor 2
building [redacted]
@[redacted]
last seen Tue, November 13, 2:12PM
room Corridor
floor 2
building [redacted]
@[redacted] Location not allowed

(b) RoamingOfficeHours

CampusBot APP 3:02 PM
"Welcome to your studygroup!"
Pinned by @CampusBot
I'm studybot and can help out with organization.
Location Ping - When at 60% of people from this studygroup are on campus, I can share their location here. Sign up below if you want me to do that.

3:02 PM Here's who's on campus on Tue, October 23, 3:02PM
@[redacted]
last seen Tue, October 23, 11:13AM
room Corridor
floor Third Floor
building [redacted]
@[redacted]
last seen Tue, October 23, 11:13AM
room Corridor
floor Third Floor
building [redacted]
@[redacted]
last seen Tue, October 23, 11:13AM
room Corridor
floor Third Floor
building [redacted]

(c) StudyGroup

Fig. 8. Sample application interface implemented as Slackbots. Users are first asked to authorize proposed pipelines. (a) the user can book rooms using Ancile; (b) Location of the user is posted to the Slack channel during certain time period; (c) Users are notified when at least 60% of them arrived on campus.

- ***BookNearMe:*** This application automatically books a convenient room based on the current indoor location of a user and the availability of nearby rooms. It tests that the pipeline abstraction is sufficient to enable the filtering of data before release to the application, and it highlights how Ancile can restrict a set of operations permitted by calendar services.
- ***RoamingOfficeHours:*** This application enables users to share location with others during prescribed hours. For example, in offices without an assigned seating plan [7]—where employees work in different locations around a building—managers can share their current location with supervisees. This application exercises the Event Subsystem of PPK and demonstrates how policies change on contextual events. We accomplish this by applying a group policy with temporal constraints.
- ***WorkGroup:*** This application helps a small group of users (e.g., students or developers) collaborate more fluidly by enabling impromptu face-to-face meetings by notifying users when a quorum of the group is on-site. It tests aggregation functions and their requirements in the PPK and DMK. We use a custom function that aggregates the location of multiple users and releases it when all of them are on-site.

Our implementation of these applications with Ancile demonstrates flexibility of use-based policies and the feasibility of creating ubiquitous computing applications with privacy awareness and enforcement. These examples are not intended as "killer apps", but rather examples of small utilities of the sort that have driven the proliferation of Slackbots and Alexa skills. Table 2 represents these three applications and the corresponding pipeline submitted to Ancile (after it is authorized as described in Section 3). We test the applications manually and report the number of submitted requests over one hour of usage. Presented results show that Ancile overhead is negligible compared to the communication overhead of fetching the applications required data from the data provider.

Table 2. Metrics and proposed pipelines for sample applications. Ancile converts each pipeline that user approves into a policy and combines it with any group policies associated with the user. It declines the pipeline if the PPK check of the group policy fails. Ancile introduces minimal overhead compare to calls associated with communicating with Location Services and Calendar Services.

Application	Book Near Me	Roaming Office Hours	Work Group
Pipeline	[*get_location, filter_data, list_rooms, book_room, return_data*]	[*get_location, return_data*]	[*get_location, when_quorum, return_data*]
Group Policy		[after_9am]	
Completed API calls	22	351	540
Rejected API calls	0	330	0
Data Provider Overhead	2374.91ms	145.54ms	300.79ms
Ancile Overhead	70.45ms	5.71ms	86.25ms

Table 3. The results of the *wrk* load tests. Each client sent the same sample request for a dummy function in Ancile and all requests were checked on a policy which allowed the dummy function.

Simulated Clients	50	100	250	500	750	1000
Mean Latency	63.67ms	128.03ms	325.97ms	652.82ms	602.03ms	717.11ms
Requests Per Second	784.85	780.48	767.95	767.55	1097.18	1100.03
Total requests	235531	234221	230462	230345	329269	330127
Request timeouts	0	0	0	116	20792	35282

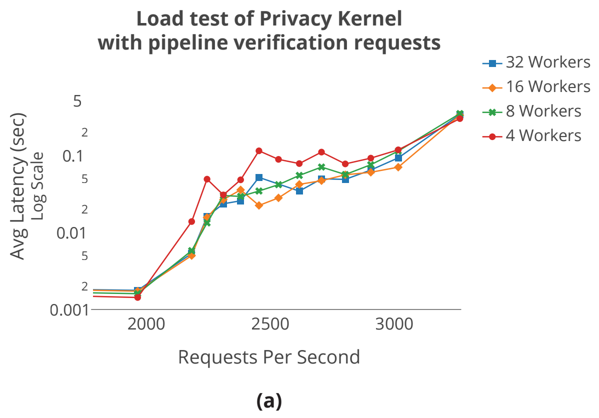
4.3 Scalability

To see how Ancile performs at scale, we conducted a series of benchmarks on the full system. The emphasis was on the Policy Kernel, to evaluate the impact of use-based privacy enforcement on system performance. Our tests include authorization round-trip latency varied on concurrent request rate and policy kernel configuration, propagation time of event triggers, and authorization round-trip time varied on pipeline length.

Load testing of Ancile. We benchmark Ancile by using the workload generation tool wrk [41] to simulate multiple clients. Each client opens a connection to Ancile and sends requests. As soon as the client receives a response from Ancile, it immediately sends another request, using the same connection. This setup models our prototype Slackbot applications that send requests to Ancile over long-lived connections. We use a request with a single dummy pipeline operation and a simple policy that authorizes the operation. This benchmark gives us an upper-bound on the number of requests Ancile can handle and how many Slackbots the system can support.

The experiment was run for five minutes with 1,000, 750, 500, 250, 100, and 50 simulated clients. The results are summarized in Table 3. It shows, that the overall system can sustain around 770 requests per second across 1 to 250 clients with acceptable performance. In the case of 750 and 1,000 simultaneous users, the number of requests overwhelms the system, leading to requests timing out, showing an artificially high number of Requests Per Second.

Privacy Policy Kernel Concurrency Load Testing. Our second experiment evaluates the performance of the PPK. Since every request needs to be authorized, understanding the performance of the PPK is paramount. To test the throughput and latency of the PPK, we run 400 simulated clients across 50 machines with each making policy



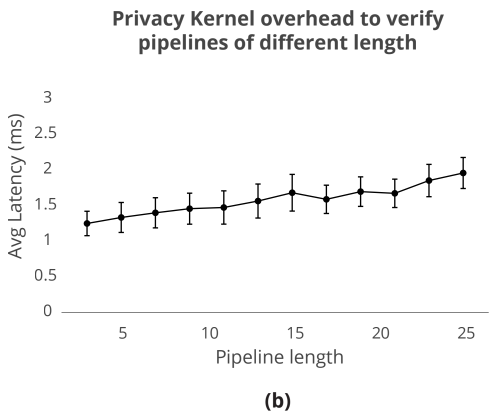


Fig. 9. Scalability benchmarks. (a) Concurrent load testing of the Policy Kernel with multiple parallel workers to process pipeline requests. The PPK can handle over 2,000 simultaneous requests with a 2 ms response time. (b) Average response time for different pipelines stays below 3 ms across the range of expected pipeline sizes.

authorization requests in the same way that authorization requests would come from the DMK. For each request, we record the round-trip time from when the request is made, to when the response from the PPK is returned. The request is the same for all clients—an authorization of a three element pipeline.

Trials are run with multiple configurations of the PPK using 4,8,16, and 32 worker processes to handle incoming requests and with request rates spanning 400 to 3,200 requests per second. Each trial was ran for 5 minutes; with statistics calculated without the leading and trailing thirty seconds of the experiment. We look at the average response time and the 95th percentile response time, as requests per second varies from 500 and 3,500 requests per second. The results are summarized in Figure 9(a). The results for 8, 16, and 32 worker threads are similar; 4 worker threads perform slightly worse. At a rate of under 2,000 requests per second, the average response time is under 1.8 ms; the 95th percentile is under 2 ms, showing a minimal overhead for requests.

At rates above 2,000 requests per second, performance starts to degrade. At a rate of 3,000 requests per second and 16 worker threads, the 95th percentile grows to 2.9 ms, but the average response time grows to 74 ms, showing that most requests complete in a reasonable amount of time, but small number of requests take an extremely long amount of time. At requests greater that 3000 per second, the latency becomes prohibitive.

Latency vs pipeline length. We evaluated how the PPK responds to authorization requests of varying complexity. We vary the length of the pipeline request, from 3 to 25 elements. For each pipeline, we send 10,000 authorization requests from a single external machine. Figure 9(b) shows the average latency varied between 1.25 milliseconds and 1.96 milliseconds, showing that our response time grows negligibly with the length of the pipeline.

Latency of event triggers. Contextual events trigger changes in policies. This final experiment evaluates how long it takes for these changes to propagate to the effected policy. We conducted 80,000 trials and observed an average propagation time of 2.2 ms from the event fired to the update being committed in the database. We consider this delay acceptable, since we anticipate system events occurring on the scale of minutes and hours, rather than milliseconds.

5 RELATED WORK

Privacy in Ubiquitous systems: User's privacy in ubiquitous applications is an important research topic [20, 23]. Sensitive data generated by ubiquitous sensors has been shown to reveal a lot of details, such as behavioral patterns and physical presence [39, 44, 45] and thus has to be treated carefully, and users will trust applications that provide stronger privacy guarantees [43]. In our experiments, we use indoor location data, because it is one of the commonly-used sensors for privacy research and it has been extensively studied over last two decades [4, 22, 28]. We use common techniques for data filtering and controlled data release to experiment with potential applications that preserve users' privacy. More advanced techniques of location obfuscation [35, 40] are not considered in this paper, but Ancile supports arbitrary functions so it is possible to design such functions.

Ancile provides an unified interface for different types of data sensors that communicate over the web and use OAuth tokens to provide authentication. As stated before, our design matches potential deployments of Ancile inside enterprises that use network infrastructure and IoT devices to create intelligent workplace environments. Our prototype applications are developed around real-life usage scenarios and can be extended to match complex systems like EasyMeeting [10]. However, we focus mostly on location privacy since it is a commonly available data source and is of significant policy concern to enterprises and end users.

Privacy Guidelines: Ancile is designed to enable application developers to comply with privacy regulations and privacy-by-design principles [23]. General Data Protection Regulation (GDPR) [38] as well as FTC guidelines [12] and OECD recommendations [30] define country-level rules and recommendations. Moreover, Institutional Review Boards [1] that approve scientific experiments require privacy measures on collected data. Large IT companies, such as Facebook, promise to create their internal versions of IRB to protect users [18]. Research studies and experimental applications now require additional resources from developers to satisfy privacy constraints, and we believe Ancile can help these developers.

Use-based privacy: As discussed in Section 2, use-based privacy [5, 6, 8, 9, 14, 27] proposes a new way to enforce data privacy and focuses restricting information use, an abstraction that has been found useful for users [6]. Our work extends the proposed approach to application developers for ubiquitous computing and Internet of Things. The theoretical framework poses a question of real-world implementation of policies and how application developers will utilize this platform. A recently proposed implementation of this framework uses SGX [5] and focuses on the enforcement mechanism of running application logic fully inside the framework, but it requires special hardware to be installed by an application or data provider. Instead, we focus here on a solution that abstracts enforcing use-based policies and allows applications to get filtered and transformed data, allowing developers to keep their software stack.

Contextual Integrity: A popular theory of privacy, Contextual Integrity [29], sets up the concept of ethical and social norms involved in information flows between parties. The framework defines roles: sender, recipient, data subject, information characteristics, and transmission principle. Ancile reflects the priorities of contextual integrity by allowing authorizations to depend on a history of contextual events, but it additionally empowers data subjects with fine-grained control of what data is requested, how that data may be used, and the conditions when data should be released. We believe that third-party app developers are interested in enforcing such types of strict data control when they need to comply with enterprise policies. For example, Apple's focus on privacy became a priority feature to enterprises such as medical systems [19], and research shows that customer's loyalty depends on privacy [11, 17]. Therefore, Ancile follows both use-based privacy and contextual integrity, and, it focuses on allowing applications to specify how data would be used and processed and provides mechanisms for control. Moreover, Ancile allows a regulator (e.g., the Principal Investigator of a study or an organization's IT security office) to set global policies for certain set of users that has to be satisfied before releasing data to third-party apps.

Privacy Preserving Systems: Many projects have focused on enforcing privacy for user data. Usage Control (UCON) [31] and Privacy Proxy [24] extend a traditional access-based approach, however, they lack full use-based policy support. Thoth [16] and Grok [33] operate on the data provider side and focus on high-performance computing. Ancile, in contrast, focuses on deployment within enterprises and assumes no changes to data provider work flow. Software Guard Extensions (SGX) [2, 13] provide additional guarantees for safe execution of programs in untrusted environments. In the current work, we don't consider SGX-based policy enforcement [5, 21, 25, 32]. Ancile is designed to be installed in a trusted environment and operates with functions written in any language, whereas SGX requires using a special SDK and C++.

Our Ancile deployment strategy is similar to Mashape marketplace [26] which abstracts APIs of multiple internet services as a single endpoint available for application developers. The model provides scalable and flexible access to multiple resources, and we use similar concept to build a layer of data processing and enforcing use-based privacy. This does raise concerns about a single point of attack, but we argue that it is easier to provide a security for a server deployed inside trusted environment then to manage every application developer or negotiate privacy enforcement with every data provider.

6 CONCLUSIONS AND FUTURE WORK

We developed Ancile to support use-based privacy for enterprise-scale ubiquitous computing applications. We focused on privacy issues involving location data, since they are highly informative to application functionality, attractive to misuse, and sensitive to the enterprise and end-user. Our initial prototype runs on commodity hardware, handles 250 simultaneous applications, and is capable of processing 750 requests per second with minimal overhead. Our example applications utilized a variety of interesting use-based privacy policies, such as aggregation and contextual events, but required minimal adaptation to run on top of Ancile. These applications are able to achieve their function using transformed information provided by Ancile instead of directly accessing sensitive enterprise and user data. This approach introduces little overhead and limits the exposure of users' data.

In addition to larger scale and longer term deployment studies, future work might explore a more systematic study of what classes of policies governing the use of location data can be expressed and enforced in Ancile, as well as additional types of data and their associated applications. In addition, there is much room for system implementation optimization, such as adjusting polling intervals to match the application's delay tolerance for delay or server load. We plan to investigate a federated version of Ancile, supporting an enterprise's ability to run multiple instances of Ancile, in future work.

REFERENCES

[1] Robert J Amdur et al. 2006. *Institutional review board: Management and function.* Jones & Bartlett Learning.

[2] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. 2013. Innovative technology for CPU based attestation and sealing. In *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, Vol. 13. ACM New York, NY, USA.

[3] Claudio Agostino Ardagna, Marco Cremonini, Ernesto Damiani, S De Capitani Di Vimercati, and Pierangela Samarati. 2007. Location privacy protection through obfuscation-based techniques. In *IFIP Annual Conference on Data and Applications Security and Privacy.* Springer, 47–60.

[4] Alastair R Beresford and Frank Stajano. 2003. Location privacy in pervasive computing. *IEEE Pervasive computing* 1 (2003), 46–55.

[5] Eleanor Birrell, Anders Gjerdrum, Robbert van Renesse, Håvard Johansen, Dag Johansen, and Fred B Schneider. 2018. SGX Enforcement of Use-Based Privacy. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society.* ACM, 155–167.

[6] Eleanor Birrell and Fred B Schneider. 2017. *A Reactive Approach for Use-Based Privacy.* Technical Report.

[7] Dianne Buckner. 2017. No desk of your own: The office plan of the future? https://www.cbc.ca/news/business/future-workplace-innovations-1.4119806.

[8] Fred H Cate. 2002. Principles for protecting privacy. *Cato J.* 22 (2002), 33.

[9] Fred H Cate, Peter Cullen, and Viktor Mayer-Schonberger. 2013. Data protection principles for the 21st century. (2013).

[10] Harry Chen, Tim Finin, Anupam Joshi, Lalana Kagal, Filip Perich, and Dipanjan Chakraborty. 2004. Intelligent agents meet the semantic web in smart spaces. *IEEE Internet computing* 8, 6 (2004), 69–79.

[11] Yu-Hui Chen and Stuart Barnes. 2007. Initial trust and online buyer behaviour. *Industrial management & data systems* 107, 1 (2007), 21–36.

[12] United States. Federal Trade Commission. 1998. *Privacy online: a report to Congress.* The Commission.

[13] Victor Costan and Srinivas Devadas. 2016. Intel SGX Explained. *IACR Cryptology ePrint Archive* 2016, 086 (2016), 1–118.

[14] Anupam Datta, Matthew Fredrikson, Gihyuk Ko, Piotr Mardziel, and Shayak Sen. 2017. Use privacy in data-driven systems: Theory and experiments with machine learnt programs. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security.* ACM, 1193–1210.

[15] Matt Duckham and Lars Kulik. 2006. Location privacy and location-aware computing. In *Dynamic and Mobile GIS.* CRC Press, 63–80.

[16] Eslam Elnikety, Aastha Mehta, Anjo Vahldiek-Oberwagner, Deepak Garg, and Peter Druschel. 2016. Thoth: Comprehensive Policy Compliance in Data Retrieval Systems.. In *USENIX Security Symposium.* 637–654.

[17] Carlos Flavián and Miguel Guinalíu. 2006. Consumer trust, perceived security and privacy policy: three basic elements of loyalty to a web site. *Industrial Management & Data Systems* 106, 5 (2006), 601–620.

[18] Goel. 2014. Facebook Promises Deeper Review of User Research, but Is Short on the Particulars Image. https://www.nytimes.com/2014/10/03/technology/facebook-promises-a-deeper-review-of-its-user-research.html.

[19] Grothaus. 2018. Forget the new iPhones: Apple's best product is now privacy. https://www.fastcompany.com/90236195/forget-the-new-iphones-apples-best-product-is-now-privacy.

[20] Jason I Hong and James A Landay. 2004. An architecture for privacy-sensitive ubiquitous computing. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services.* ACM, 177–189.

[21] Tyler Hunt, Zhiting Zhu, Yuanzhong Xu, Simon Peter, and Emmett Witchel. 2016. Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data.. In *OSDI.* 533–549.

[22] John Krumm. 2009. A survey of computational location privacy. *Personal and Ubiquitous Computing* 13, 6 (2009), 391–399.

[23] Marc Langheinrich. 2001. Privacy by designâĂŤprinciples of privacy-aware ubiquitous systems. In *International conference on Ubiquitous Computing.* Springer, 273–291.

[24] Marc Langheinrich. 2002. A privacy awareness system for ubiquitous computing environments. In *international conference on Ubiquitous Computing.* Springer, 237–245.

[25] Joshua Lind, Christian Priebe, Divya Muthukumaran, Dan O'Keeffe, P Aublin, Florian Kelbert, Tobias Reiher, David Goltzsche, David Eyers, Rüdiger Kapitza, et al. 2017. Glamdring: Automatic application partitioning for Intel SGX. USENIX.

[26] mashape [n. d.]. Mashape - Free API Management Platform and Marketplace. https://market.mashape.com/.

[27] Craig Mundie. 2014. Privacy Pragmatism; Focus on Data Use, Not Data Collection. *Foreign Aff.* 93 (2014), 28.

[28] Ginger Myles, Adrian Friday, and Nigel Davies. 2003. Preserving privacy in environments with location-based applications. *IEEE Pervasive Computing* 1 (2003), 56–64.

[29] Helen Nissenbaum. 2004. Privacy as contextual integrity. *Wash. L. Rev.* 79 (2004), 119.

[30] OECD. 2002. *OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data.* Organisation for Economic Co-operation and Development.

[31] Jaehong Park and Ravi Sandhu. 2002. Towards usage control models: beyond traditional access control. In *Proceedings of the seventh ACM symposium on Access control models and technologies.* ACM, 57–64.

[32] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. 2015. VC3: Trustworthy data analytics in the cloud using SGX. In *Security and Privacy (SP), 2015 IEEE Symposium on.* IEEE, 38–54.

[33] Shayak Sen, Saikat Guha, Anupam Datta, Sriram K Rajamani, Janice Tsai, and Jeannette M Wing. 2014. Bootstrapping privacy compliance in big data systems. In *Security and Privacy (SP), 2014 IEEE Symposium on.* IEEE, 327–342.

[34] Company Data Sheet. [n. d.]. ARUBA 310 SERIES ACCESS POINTS. https://www.arubanetworks.com/assets/ds/DS_AP310Series.pdf.

[35] Reza Shokri, George Theodorakopoulos, and Carmela Troncoso. 2017. Privacy games along location traces: A game-theoretic framework for optimizing location privacy. *ACM Transactions on Privacy and Security (TOPS)* 19, 4 (2017), 11.

[36] slack [n. d.]. Slack API, Slack. https://api.slack.com.

[37] t2micro [n. d.]. Amazon EC2 T2 Instances. https://aws.amazon.com/ec2/instance-types/t2/.

[38] Paul Voigt and Axel Von dem Bussche. 2017. *The EU General Data Protection Regulation (GDPR).* Vol. 18. Springer.

[39] Hao Wang, Daqing Zhang, Junyi Ma, Yasha Wang, Yuxiang Wang, Dan Wu, Tao Gu, and Bing Xie. 2016. Human respiration detection with commodity wifi devices: do user location and body orientation matter?. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing.* ACM, 25–36.

[40] Leye Wang, Dingqi Yang, Xiao Han, Tianben Wang, Daqing Zhang, and Xiaojuan Ma. 2017. Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation. In *Proceedings of the 26th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 627–636.

[41] wrk [n. d.]. Modern HTTP benchmarking tool. https://github.com/wg/wrk.

[42] Jie Xiong and Kyle Jamieson. 2013. ArrayTrack: a fine-grained indoor location system. Usenix.

[43] Heng Xu, Na Wang, and Jens Grossklags. 2012. Privacy by redesign: Alleviating privacy concerns for third-party apps. (2012).

[44] Daqing Zhang, Hao Wang, and Dan Wu. 2017. Toward centimeter-scale human activity sensing with Wi-Fi signals. *Computer* 50, 1 (2017), 48–57.

[45] Yongpan Zou, Weifeng Liu, Kaishun Wu, and Lionel M Ni. 2017. Wi-fi radar: Recognizing human behavior with commodity wi-fi. *IEEE Communications Magazine* 55, 10 (2017), 105–111.