A Privacy Negotiation Mechanism for IoT

Khaled Alanezi
Computer Department
College of Basic Education, PAAET, Kuwait
kaa.alanezi@paaet.edu.kw

Shivakant Mishra
Computer Science Department
University of Colorado at Boulder, CO, USA
mishras@cs.colorado.edu

ABSTRACT

This paper presents a new privacy negotiation mechanism for an IoT environment that is both efficient and practical to cope with the IoT special need of seamlessness. This mechanism allows IoT users to express and enforce their personal privacy preferences in a seamless manner while interacting with IoT deployments. In addition, the proposed mechanism satisfies the privacy requirements of the IoT deployment owner. Finally, the proposed privacy mechanism is agnostic to the actual IoT architecture and can be used over a user-managed, edge-managed or a cloud-managed IoT architecture. Prototypes of the proposed mechanism have been implemented for each of these three architectures, and the results show the capability of the protocol to negotiate privacy while adding insignificant time overhead.

1 INTRODUCTION

The Internet of Things (IoT) as a new revolutionary vision in computing has emerged recently. An IoT infrastructure is characterized by millions of devices or simply things that are connected to the Internet. A myriad of Interesting applications can be built by enabling users to interact with this infrastructure while navigating through spaces in their daily life. Although this vision of IoT is not yet fully utilized, it is typical nowadays to encounter different types of IoT deployments in public places as well as in private households. In addition, open communication standards are being investigated to communicate for these deployments that enable users to access both IoT sensors and actuators. A remaining challenging question is how to protect users' privacy given the ubiquitous nature of IoT deployments? Since traditional privacy settings and privacy notice/choice screens are not applicable as they hinder the seamless experience of the IoT, it is important to come with a privacy negotiation mechanism that is both efficient and practical to cope with this IoT special need of seamlessness.

In this paper, we address this challenge by proposing a privacy negotiation mechanism to allow IoT users to express and enforce their personal privacy preferences while interacting with IoT deployments. On one hand, the negotiation mechanism is designed to be practical by allowing the user device to negotiate with the IoT deployment in the background on behalf of the user. By using this scheme users can control data collection activities of IoT deployments about them without the burden of accessing the device to fill setting screens or read a privacy notice. On the other hand, this privacy negotiation mechanism is efficient as it is built on top of existing IoT communication protocols and adds only a negligible overhead to the ongoing communication of IoT applications. A key novelty of this work comes from demonstrating through practical experiments the viability of automated privacy negotiation over IoT architectures.

It is important that the proposed negotiation mechanism not only covers the privacy requirements of the IoT users (i.e. IoT service consumers) but extends this coverage to negotiate and satisfy the privacy requirements of the IoT deployment owner (henceforth IoT owner) as well. The IoT owner is the responsible party for setting up and maintaining the IoT infrastructure that provides services to IoT users. Generally, providing services involves enabling the IoT user to access sensory interfaces such as temperature, audio or camera sensors. We envision that the IoT owners would like to manage this access according to their privacy requirements since every access to a sensor can be used to reveal information about them. To cover the privacy requirements of IoT users and owners, we model the problem as a utility-privacy tradeoff function in which sharing more information increases the utility gained from the service but leads to more undesired privacy exposure. We show that the negotiation protocol can use multiphase negotiation efficiently to embark into an agreement that satisfies the utility and privacy requirements of IoT users and owner simultaneously.

Another important aspect that must be covered by the negotiation protocol is the handling of current diverse IoT infrastructures. IoT infrastructures are envisioned to act as utility network for IoT users to use IoT services on the go [12]. However, the gateway to access these services can be either a central server, cloud server or edge server, or the infrastructure may be accessed directly using M2M communication. An important feature of our negotiation protocol is that it is architecture agnostic and can be used on cloud based, edge-based or M2M types of infrastructure.

In summary, our contributions are as follows:

- We have designed a negotiation protocol to model and realize privacy in an IoT ecosystem. The protocol takes a holistic approach by covering the requirements of all involved parties in an IoT interaction. The practicality of the protocol stems from modeling privacy as a tradeoff function with the

utility achieved from using/providing IoT services.
• We have developed and tested a prototype of the proposed protocol over three different IoT infrastructure standards of user-managed, edge-managed and cloud-managed IoT architectures. Results show the capability of the protocol to negotiate privacy while adding insignificant time overhead.

2 SCENARIOS

We identify three IoT scenarios that are common in literature namely user-managed IoT, edge-managed IoT and cloud-managed IoT. In a user-managed IoT architecture, the IoT owner's mobile device, e.g. a smartphone or a tablet acts as a gateway to the IoT infrastructure. All negotiation and access must happen through this mobile device. This is analogous to an IoT infrastructure installed at home or a private office. Consider a smarthome environment where a home is equipped with a range of sensors such as a temperature sensor, a carbon monooxide sensor, etc. These smarthome sensors are designed to operate in a star topology network, where each sensor is connected to a single gateway, e.g. home owner's smartphone, and sends its sensor data to that gateway at regular intervals. In this case, if another device, e.g. a utility person's smartphone needs to access this sensor data, possibily for a limited time, the sensor data must be retrieved from the home owner's smartphone. In this scenario, the home owner's smartphone must be involved in privacy negotiation with the utility person's smartphone.

For edge-managed and cloud-managed IoT, a server plays the role of the gateway and carries all negotiation and access requests. For the former, the server is within the same network domain for the IoT infrastructure[5]. Whereas, for the latter, the server is situated in a public cloud. For example, in a smartcity, video feed from a security camera may be stored at an edge server or a cloud server. If an agency such as a law enforcement agency needs to access this video feed, the egde or cloud server must be involved in privacy negotiation with the law enforcement agency.

3 DESIGN

Privacy Requirements

The fact that the negotiation protocol operates in IoT environments imposes stringent practicality requirements on our design. We describe below three of these requirements that must be satisfied by the protocol design.

(1) **No User Involvement:** All negotiation communication must take place in the background and without user intervention. When users navigate through public or private spaces, they will typically encounter IoT deployments (i.e. IoT owners) and engage in a data exchange with them to utilize a particular service. Due to the difficulty in involving the IoT user in such situation, the IoT user's device that has the user privacy requirements should act seamlessly to negotiate these privacy requirements on behalf of the user with the IoT owner. Also, the IoT owner's privacy settings will be communicated to the IoT user to ensure their adherence to these requirements.
(2) **Minimal Overhead:** The negotiation protocol must impose minimal time and energy overhead to the IoT task. IoT services are typically provided promptly to the users. Therefore, these services are sensitive to any time delays, which should be considered in the protocol design. Also, since IoT devices are battery powered, the protocol must be energy efficient to avoid draining the energy sources of involved devices.
(3) **Choice Flexibility:** The protocol should avoid the current privacy notice and choice model of either accepting the service as a whole or abandoning it as we see that this model is rigid and will not work with IoT situations requiring more flexibility. Diverse options should be offered to IoT users to enable them to continue to use the service while not sacrificing their privacy requirements.

Privacy Model

The negotiation protocol presented here is in alignment with the vision of achieving openness in IoT environments. Openness envisions IoT networks to act as a utility infrastructure, similar to electricity and water, that is accessed by IoT users on the go [12]. Open environments as such requires that the parties involved in information exchange specify their privacy requirements to be negotiated on their behalf. This is analogous to the P3P protocol [3] in which a browser negotiates the privacy requirements of users on their behalf with visited websites to control the personal information that the website can collect about the user. In our protocol, the IoT user and IoT owner will store their privacy requirements in a policy file stored locally and written using XML language. An example scenario for privacy policies for an IoT owner and an IoT user is shown in Listing 1 and Listing 2 respectively.

Listing 1: IoT User Privacy Policy

```
<privacy-policy>
  <data-in type="image" priority="1">
    <retention>3-month</ retention >
    <shared>no</shared>
    <inferred>yes</inferred>
  </data-in>
  <data-out>
  <data-out type="video" priority="1">
    <retention>1-year</ retention >
    <shared>no</shared>
    <inferred>no</inferred>
  </data-out>
</privacy-policy>
```

Listing 2: IoT Owner Privacy Policy

```
<privacy-policy>
  <data-in type="video" priority="1">
    <retention>1-year</ retention >
    <shared>no</shared>
    <inferred>yes</inferred>
  </data-in>
  <data-out type="face-detection" priority="1">
    <retention>1-year</ retention >
    <shared>no</shared>
    <inferred>no</inferred>
  </data-out>
  <data-out type="image" priority="1">
    <retention>1-year</ retention >
    <shared>no</shared>
    <inferred>yes</inferred>
  </data-out>
</privacy-policy>
```

As seen in Listing 1, the privacy policy for the IoT user specifies **<data-in>** tags indicating the type of data that the user would like to acquire from the IoT owner along with child elements specifying the usage scenario for this data. These **<data-in>** tags from the IoT user privacy policy will be matched against the **<data-out>** tags in the IoT owner's policy since the latter specifies the data collection practices accepted by the IoT owner. Conversely, the **<data-in>** tags specified in the IoT owner's policy in Listing 2 will be matched against the **<data-out>** tags in the IoT user privacy policy to ensure that the level of data collection performed by the IoT owner is acceptable by the IoT user. An interesting research question is how to learn the privacy policy for each user which could be different for different locations? Other research [8] has shown that the privacy preferences of a user can be predicted by observing few data collection scenarios. Using prediction is beneficial to avoid the cumbersome and error prone task of filling privacy settings screens. One of our key design requirements is achieving flexibility by allowing the protocol to negotiate multi-levels of service with different scales of data collection scenarios. This is required due to the high cost of an unsuccessful negotiation as it might require the user to leave the place to avoid the data collection process altogether. To address this challenge, the negotiation protocol models the relationship between data collection and the IoT service as a utility-privacy tradeoff function stated as Formula 1. This allows the IoT owner to offer multiple choices of service levels, measured by the utility, to the IoT user based on how much data they are willing to share? We choose to employ four dimensions form factors influencing privacy preferences in IoT environments from [8] in this utility-privacy tradeoff function. These four factors are saved in the privacy policy XML representation as child elements inside each **<data-in>** and **<data-out>** tags. The description for each of these four elements is as follows:

(1) **Data Type (***t***).** The type of sensor being accessed can have varying degree of exposure to the privacy of the owner of that sensor. Sensors such as the camera or the microphone are inherently sensitive. Hence, allowing access to those sensors must be handled with care. There are techniques in literature to minimize the degree of privacy exposure when accessing those sensors such as blurring faces from the live video feed of a surveillance camera [4] or carefully choosing audio features to avoid construction of speech from captured audio data [13]. If used, these techniques must be added to the XML file to be part of the negotiation process.
(2) **Retention. (***r***)** Retention policy specifies time durations for keeping logs of exchanged data. In real-time applications, where no data storage is required, this factor can be used by the IoT owner or the IoT user to enforce purging their data by leaving this element empty.
(3) **Shared (***s***).** Any third party recipient must be specified in case the IoT owner or the IoT user is sharing the any gathered data for the IoT task.
(4) **Inferred (***i***).** The recipient of the data must specify if inference techniques will be used to gain further information from the data. For example, accelerometer data can be used to monitor exercising habits of a user for health applications but can also be used in dead reckoning techniques [10] to determine indoor user location.

Other form factors that can also be considered include the location, purpose and the benefit of the data collection [8]. After learning the aforementioned privacy influencing factors in the negotiation exchange described next subsection, both the IoT owner and IoT user will use them as part of a privacy-utility calculation to ensure that the achieved utility from the IoT service outweighs the degree of privacy exposure. We adopted the privacy-utility function in [9], which serves e-commerece sites, with modifications to fit to IoT applications scenario. This function is used by both the IoT user and the IoT owner of the data to evaluate the privacy-utility of the the data exchanged between them. The utility-privacy function is as follows:

$$U = -\gamma . P_e(t, r, s, i) + B(t, r, s, i) \tag{1}$$

Explaining the notations used in the formula:

U denotes the total utility that will be achieved from pursuing the information exchange to run the IoT application.

B is the benefit from the data exchange as seen from the perspective of the data owner. For the IoT owner, this could be a monetary incentive or the social benefit from allowing IoT applications to run on their premises. As for the IoT user, the benefit would be the service provided by the IoT application. Notice that B is a function of privacy exposure form factors as we expect the benefit to be proportional to the selected policy for the data item.

P_e is the degree of privacy exposure for the selected privacy policy. Different privacy policies will lead to different levels of privacy exposure based on the chosen exposure form factors chosen in the policy. For example, higher retention periods specified by r for highly sensitive data specified by t will lead to higher values of P_e.

Notice that the exact function for B and P_e can be as simple as sum or product for predefined values reflecting the degree of influence for each of the privacy factors. However, choosing a personalized function to fit to the user's own preference is out of the scope of this paper.

γ is an overall privacy sensitivity perception factor. This factor can vary depending on the location or context of the user. For example, users might be comfortable for taking a picture for them in public places as opposed to being in private places. γ is multiplied by P_e to either escalate or deescalate the total privacy leakage for the specific data sharing situation based on the context and/or location.

Note that the product term to the right of the equality operator is negative so as to reconcile it with the benefit term B. Thus, the utility U will be positive when the value of the benefit B term outweighs the value of the negative privacy exposure term and vice versa.

In summary, the presented privacy model uses XML to store the privacy policies and use them without user intervention. Also, it supports the requirement of choice flexibility by modeling the utility of the service as a function of privacy exposure form factors. Section 5 demonstrates that the model and the overall protocol satisfies the minimum overhead requirement when implemented over well know IoT architectures.

Privacy Negotiation

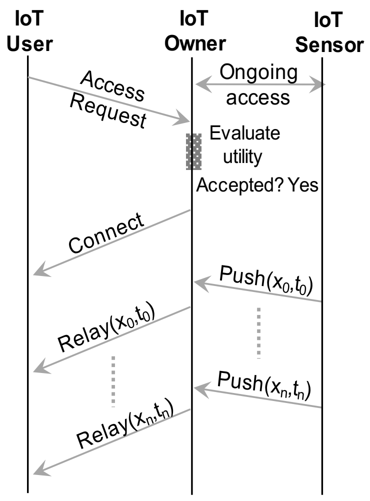
This section describes the flow of the negotiation activities between the IoT user and the IoT owner. We have implemented and evaluated the privacy negotiation protocol on all three architecture scenarios described in Section 2. The design of the negotiation protocol is made flexible by avoiding the go/no-go scheme of current privacy/notice choice mechanisms and replacing it with multi-phase negotiation. This section describes 1-phase negotiation and 2-phases negotiation. We start by describing the 1-phase negotiation scenario shown in Figure 1 (a). The scenario starts with the IoT user sending an access request for specific type of data (i.e. sensor) to the IoT owner. The IoT user will embed a summary of their usage requirements for this data, which is taken from the **<data-in>** element for this type of data in the user policy. Upon receiving the access request, the IoT owner checks the utility of the request by substituting it in the utility function. Assuming that the utility of the request is equal or higher than the utility achieved when substituting the **<data-out>** element for the same data from the IoT owner privacy policy, the owner accepts the request. After that, the IoT owner connects to the IoT user and starts acting as a relay by forwarding sensor information received from the IoT infrastructure to the IoT user. The 2-phases negotiation is shown in Figure 1 (b). This scenario starts in a similar way by the IoT user sending an access request to the IoT owner. However, in this scenario the utility of the request is deemed unacceptable by the IoT owner. Accordingly, the IoT owner will retrieve the **<data-out>** element for the requested data item from their own privacy policy, which represents their acceptable usage scenario for the data item and send it as a proposal to the IoT user. This only happens after connecting to the IoT user. The IoT user's device then checks the utility of this proposal against a second priority policy if one is defined in their privacy policy file. Note that each **<data-in>** and **<data-out>** tag in the privacy policy contains a priority attribute to allow the IoT user and IoT owner define alternative policies to be used during negotiations. Only defining a priority 1 policy for the data item means that the policy for this data item is non-negotiable. Assuming the IoT user has accepted the alternative proposal, the IoT owner will start forwarding required data as soon as it is received from the data source.

4 IMPLEMENTATION

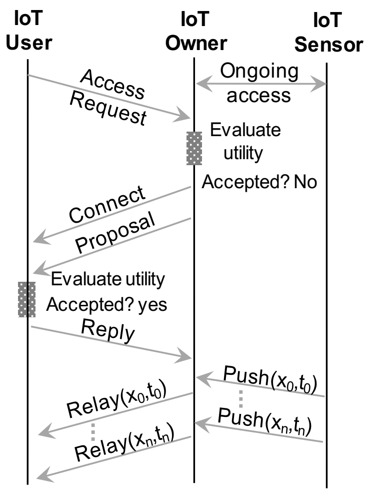
In a user-managed architecture, the dominant communication protocol is BLE to conserve energy, while for the edge-managed and cloud-managed architectures, a server acts as a gateway and communication is done using WiFi. Hardware used to simulate these infrastructures is shown in Table 1.

User-Managed IoT Experiment

The picture in Figure 2a demonstrates the setup for the user-managed IoT experiment. We used the breadboard to connect a temperature sensor to the Arduino Uno. Then, we enabled BLE connectivity functionality on the Arduino board by connecting it to an Adafruit Bluefruit BLE shield through the breadboard. Using the Arduino IDE, a code was written to enable the Arduino board to broadcast using BLE beacons its availability in order for BLE clients to connect to it and read the temperature sensor values. The two Android-based Moto E smartphones were then used to simulate the IoT owner and IoT user mobile devices. We wrote Android code to enable the IoT owner to connect to the Arduino board and start receiving temperature values as push notifications periodically. Meanwhile, the IoT owner registers a BLE search to listen to any guest devices that might need a sensor reading from the IoT environment they own. Note here that the BLE



(a) 1-Phase Negotiation

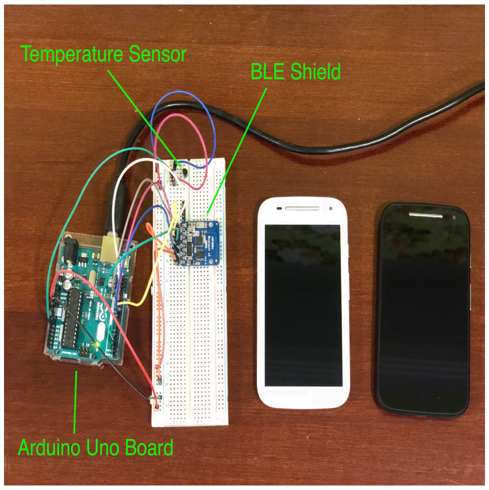


(b) 2-Phases Negotiation

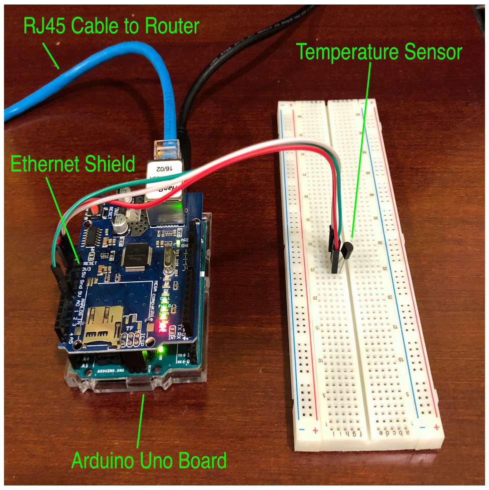
Figure 1: Privacy Negotiation Flow

Table 1: Hardware Testbed

Title	Function	Scenario Used
Arduino Uno R3	Microcontroller for the IoT device	Both
LM35 Temperature Sensor	Sensory interface to be accessed	Both
Arduino Ethernet Shield	Adds ethernet connectivity to the Arduino Uno	Edge-managed
Arduino BLE Shield	Adds BLE connectivity to the Arduino Uno	User-managed
D-Link Wireless Router	Wireless LAN router	Edge-managed
Motorola Moto E smartphone	IoT user requesting access to a sensor	Both
Motorola Moto E smartphone	IoT owner	User-managed
MacBook Air	Computing node at the edge of the network	Edge-managed
Amazon EC2 Micro Instance	Computing node in the Cloud	Cloud-managed



(a) User-managed experiment.



(b) Cloud\Edge-managed experiment.

Figure 2: Hardware Setup.

search is registered under a specific UUID designated for the sensor sharing services. Searching for specific UUID using BLE can happen in the background while the device is in sleep mode thereby drastically reducing energy consumption. This means that the cost of detecting collaborators for the IoT owner is trivial.

Upon arriving to an environment an IoT owner who is looking for a type of sensor reading to perform a particular service sends a BLE advertisement broadcasting its intention to access a shared sensor. This BLE broadcast contains the UUID for the sharing service and a vector containing the request information as described in Section 3. The request information here is a summary from the **<data-in>** tags describing the data that the device requires along with the usage scenario. This broadcast will be captured by the IoT owner as it contains the UUID of the sharing service. In case the IoT owner accepts the request, it will connect to the requesting device and starts relaying temperature readings to it as soon as they arrive from the Arduino board. Otherwise, as described in Section 3, the negotiation flow requires a second round of negotiation. In this case, the IoT owner's device will connect to the IoT user's device and send a proposal containing it's acceptable privacy policy for the required sensor. The requester can now either accept or reject this proposal. The requester will use the ongoing BLE connection to reply. If the reply is accept, the owner will start relaying temperature values to the IoT user. Notice that in this situation the IoT owner is a hub for a BLE star topology network with the IoT user and the Arduino board acting as the hosts. We demonstrate the time efficiency for the communication in the evaluation in Section 5. If the IoT user rejected the proposal, the owner will simply teardown the BLE connection and continue operating normally.

Edge-Managed and Cloud-Managed IoT Experiments

We also show in Figure 2b the setup for the edge-managed and cloud-managed IoT experiments.The Ethernet shield is stacked on top of the Arduino board to provide it with Ethernet capability. After that, an Ethernet cable is used to connect the Arduino board to the wireless LAN router. This allows the Arduino to get a local IP and is now able to communicate with other devices within the same wireless LAN. We used the Arduino IDE to write code to let the Arduino Uno acts as web server providing an HTML page to read the temperature sensor value. This setup is then used to perform 1-phase negotiation and 2-phases negotiation for the edge-managed and cloud-managed IoT scenarios. For the former, we used a MacBook Air laptop to act as an edge server by running Java code listening to network communication at specific port within the same LAN. For the latter, the same Java code was deployed to an Amazon EC2 instance that is used as a cloud server. Then, port forwarding was used on the wireless LAN router to enable communication between the cloud server and the Arduino board over the Internet. For both scenario, the server (i.e. IoT owner) receives a request from a mobile device resembling the IoT user for sensory data. This request contains a summary from the IoT user policy. The server will answer with the sensor information or with a proposal in case further negotiation is required. The client will either accept or reject this proposal. If the proposal was accepted, the server will perform an HTTP GET to get the temperature and send the result.

5 EVALUATION

User-Managed IoT

The negotiation protocol in user-managed IoT utilizes BLE to communicate and serve requests for data sharing. A star network is formed whenever an acceptable request arrives from a IoT user in which the IoT owner becomes the central node and the IoT user and any future IoT users will become host nodes. The complexity and time\energy efficiency of this process are reported in our previous work [1]. Note that the IoT owner can serve simultaneous IoT users by simply joining new users to this BLE star network. Also note that we choose to implement this communication mechanism using BLE as it is becoming a standard communication protocol for IoT devices. Nevertheless, the negotiation protocol can be implemented over other communication standards such as ZigBee or NFC. Figure 3 shows the aggregate time taken for every phase in the negotiation. The aggregate time is calculated from the beginning of the request (i.e. sending BLE broadcast embedded with data request information) and involves the time taken for the previous phases. We performed each experiment five times and report the average with the standard error on each bar. The figure shows that, after broadcasting a request in a BLE beacon, it takes around 800 ms to be connected to an IoT owner in the place who is willing to support this request. Recall from the design section that after establishing a connection two situations might occur. First, The IoT owner accepts the utility of the request and sends the required sensor data in what we call 1-phase negotiation. The total time for this situation is reported in Figure 5 along with the total time for the other two scenarios of no-privacy and 2-phases negotiation. Second, the IoT owner might offer an alternative proposal that is suitable for them. The total time for the IoT user to receive the alternative proposal is 1400 ms on average. We note here that the additional time of 600 ms is dominated by the time required by the BLE to interrogate the IoT user's device after connecting to it to be able to call its services. Assuming that the proposal is accepted, replying to the proposal and receiving the sensor value requires an additional 30 ms only. Figure 4 reports aggregate time for the negotiation protocol

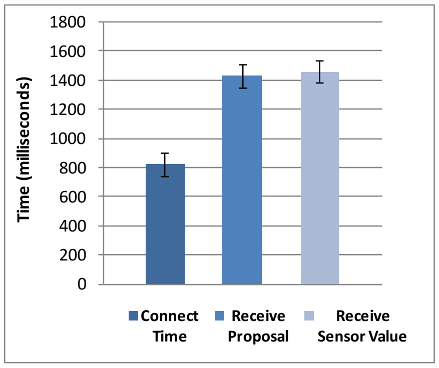


Figure 3: Aggregate time for major negotiation milestones (IoT User\User-Managed IoT.)

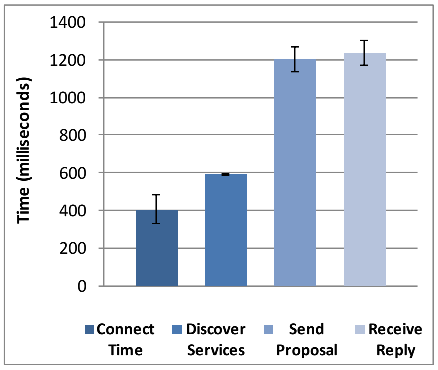


Figure 4: Aggregate time for major negotiation milestones (IoT Owner\User-Managed IoT.)

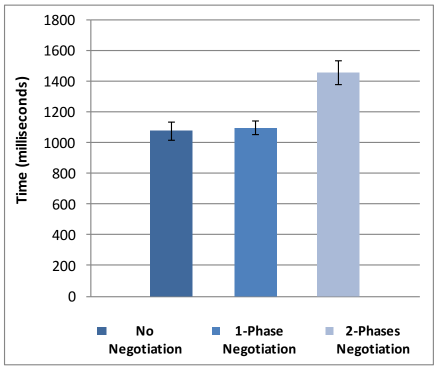


Figure 5: Total time to receive sensor reading for no-negotiation, 1-phase & 2-phases negotiation in user-managed IoT.

from the IoT owner's perspective. The average total time to connect to a device after detecting a supported request in a BLE broadcast is 400 ms. This is followed by an almost 600 ms for discovering services on the IoT user's device. We report the connect time and the service discovery separately. The time taken from the moment a beacon is detected to sending an alternative proposal is around 1200 ms on on average. It takes an additional 30 ms on average for the IoT owner's device to receive a reply to their proposal. Finally, we report in Figure 5 the average overall time to receive the sensor data for 1-phase communication and 2-phases communication. This time is measured from the moment the IoT user sends a beacon with embedded request information to the moment they actually receives the required sensor data (i.e. the temperature sensor reading). We also include a no-negotiation scenario in which the sensor data is sent to the requester immediately without matching the specification of the request with the privacy requirements of the IoT owner. As expected, the results show that the 1-phase negotiation adds negligible time overhead compared to the negotiation scenario as it only adds information to the beacon and process them at the IoT owner's side. On the other hand, the 2-phases negotiation adds on average 350 ms, which includes the time required to receive an alternative proposal from the IoT owner, reply by accepting the proposal, then receiving the sensor value.

Edge-Managed and Cloud-Managed IoT

In this subsection, we repeat the experiments in the previous subsection but with the IoT now being managed by either an edge or a cloud server. First, we begin by describing the results of the edge-managed scenario. The IoT user needs to negotiate with and access the IoT infrastructure through the edge server and all communication is happening over wireless LAN network. Figure 6 shows the major milestones from the IoT user's side in this negotiation situation. We note two things in this experiment. First, as opposed to the user-managed negotiation scenario, there is no connect phase since the IoT owner is now a server that continuously listens to a network port to reply to requests from IoT users. Hence, there is no connect phase per se. Second, unlike the user-managed IoT experiment, we don't report in this experiment results from the IoT owner's side since the owner is now a server with presumably abundant resources. We see from the figure that the time to receive an alternative proposal from the IoT owner is now less than 400 ms. This time includes the time to send a request to the edge server with an unacceptable privacy settings and receiving back an alternative proposal. Furthermore, the time to receive the sensor value if the IoT user accepted the IoT owner's proposal is around 500 ms. We conclude that managing the IoT infrastructure using an edge server provided much better time efficiency than the user-managed IoT scenario using BLE. We also report in Figure 7 a comparison between three situations for negotiating privacy requirements. There are two important lessons that can be learned from this figure. First, when comparing the no-negotiation scenario with the 1-phase negotiation scenario, we see a difference of around 20 ms. This difference is negligible as it is attributed to the variability of the performance of the wireless LAN, which can be realized from the standard errors. Therefore, similar to the results from the user-managed IoT, these two situations have similar time performance. Second, we see that adding a second negotiation phase added an average of 200 ms attributed to sending the proposal from the IoT owner and receiving a reply back before sending the required sensor reading. Overall, the performance is still better than the same scenario for the user-managed IoT. Finally, Figure 8 reports similar results for the previous figure but with a cloud-based server now used to manage IoT as opposed to

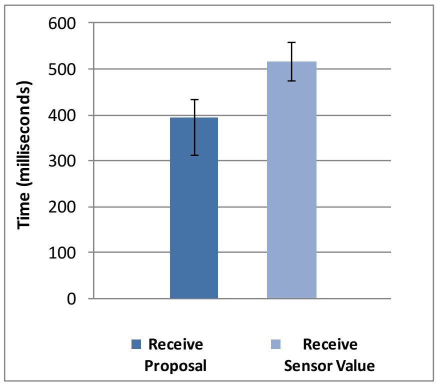


Figure 6: Aggregate time for major negotiation milestones (IoT User\Edge-Managed IoT.)

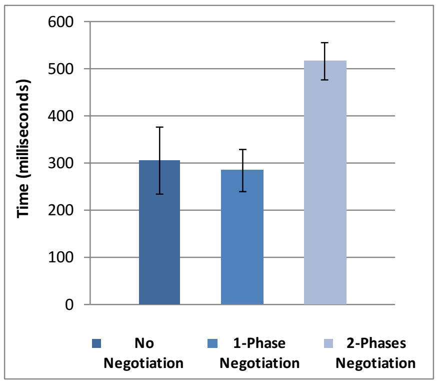


Figure 7: Total time to receive sensor reading for no-negotiation, 1-phase & 2-phases negotiation in edge-managed IoT.

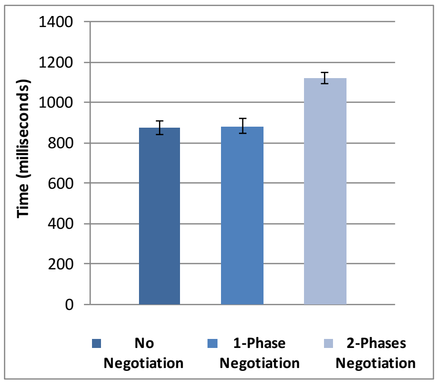


Figure 8: Total time to receive sensor reading for no-negotiation, 1-phase & 2-phases negotiation in cloud-managed IoT.

an edge-based server. We notice from the figure that the latency doubled in all three scenarios due the need of routing the communication through the Internet. This result is in harmony with other research [11] that highlighted the benefits that cloudlets proximity brings for better management of IoT infrastructure but highlighted various challenges for this paradigm to become a reality.

6 RELATED WORK

There is a consensus among researchers that difficulty in preserving user privacy is a major hurdle for wide adoption of IoT. E.g. [8, 12]. The authors in [7] presented a framework that allows the user to easily define their privacy requirements and enforce them at check points in the network. The privacy coach [2] is another work aimed at protecting user privacy in IoT that is geared towards RFID technology. It follows the same approach of using an application running on a mobile device to mediate between the user and the deployment owner. The authors in [6] utilized Blockchain technology to protect users from attacks on their privacy in IoT environments. The position paper in [5] presented an architecture based on edge computing to allow the user to control access to their data in an IoT system.

7 CONCLUSION

This paper presented a privacy negotiation scheme to address the privacy requirements of users in IoT environments. The proposed approach is practical as it negotiates the privacy policy of the user with the IoT owner without user intervention and supports the selection from among multiple predefined IoT user and owner privacy policies. The feasibility of the negotiation protocol was demonstrated by means of a thorough implementation and evaluation over three widely accepted IoT scenarios.

8 ACKNOWLEDGEMENTS

The first author would like to thank the Kuwait Foundation for the Advancement of Science for financially supporting his trip to present this paper at the IEEE DASC 2018 conference.

REFERENCES

[1] K. Alanezi, R. Rafiq, L. Chen, and S. Mishra. 2017. Leveraging BLE and social trust to enable mobile in situ collaborations. In *IMCOM*.
[2] G. Broenink, JH Hoepman, C. Hof, R. Van Kranenburg, D. Smits, and T. Wisman. 2010. The privacy coach: Supporting customer privacy in the internet of things. *arXiv preprint arXiv:1001.4459* (2010).
[3] L. Cranor. 2002. *Web privacy with P3P.* " O'Reilly Media, Inc.".
[4] A. Das, M. Degeling, X. Wang, J. Wang, N. Sadeh, and M. Satyanarayanan. 2017. Assisting Users in a World Full of Cameras: A Privacy-Aware Infrastructure for Computer Vision Applications. In *CVPRW*. IEEE, 1387–1396.
[5] N. Davies, N. Taft, M. Satyanarayanan, S. Clinch, and B. Amos. 2016. Privacy mediators: Helping iot cross the chasm. In *HotMobile*. ACM, 39–44.
[6] A. Dorri, SS Kanhere, R. Jurdak, and P. Gauravaram. 2017. Blockchain for IoT security and privacy: The case study of a smart home. In *PerCom*. IEEE, 618–623.
[7] M. Henze, L. Hermerschmidt, D. Kerpen, R. Häußling, B. Rumpe, and K. Wehrle. 2014. User-driven privacy enforcement for cloud-based services in the internet of things. In *FiCloud*. IEEE, 191–196.
[8] PE Naeini, S. Bhagavatula, H. Habib, M. Degeling, L. Bauer, L. Cranor, and N. Sadeh. 2017. Privacy Expectations and Preferences in an IoT World. In *SOUPS*.
[9] Sören P. 2006. Implementing privacy negotiations in e-commerce. In *Asia-Pacific Web Conference*. Springer, 604–615.
[10] AR Pratama, R. Hidayat, et al. 2012. Smartphone-based pedestrian dead reckoning as an indoor positioning system. In *ICSET*. IEEE, 1–6.
[11] M. Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (2017), 30–39.
[12] JA Stankovic. 2014. Research directions for the internet of things. *IEEE Internet of Things Journal* 1, 1 (2014), 3–9.
[13] D. Wyatt, T. Choudhury, and J. Bilmes. 2007. Conversation detection and speaker segmentation in privacy-sensitive situated speech data. In *Interspeech*.