# The Case for Energy-aware Trust Establishment in Dynamic Networks of Cyber Physical Devices

Amruta Gokhale, John McCabe, Vinod Ganapathy, Ulrich Kremer

Rutgers University
{amrutag,jomccabe,vinodg,uli}@cs.rutgers.edu

**Abstract.** In a dynamic network, cyber physical devices collect and share data by opportunistically connecting with each other. By their very nature, computations in dynamic networks may be distributed among several untrusted devices, some of which may be malicious in intent. It is therefore key to have mechanisms that allow one device to establish the trustworthiness of another device in the dynamic network.
It is possible to address this problem using trusted computing protocols proposed in the research literature. However, in this paper we make the case that existing trusted computing protocols are not directly applicable to dynamic networks. This is because cyber physical devices in a dynamic network are often resource-constrained, and trusted computing protocols involve challenges/responses between multiple communicating parties, which may lead to a significant strain on these devices' limited resources. We therefore conclude with a call for research on energy-aware trust establishment for dynamic networks.

**Keywords:** dynamic networks; trusted computing; attestation protocols; energy consumption; resource-aware protocols.

## 1 Introduction

Dynamic networks allow cyber physical devices to spontaneously connect and share data using different communication infrastructures, including fixed and ad hoc networking. This paper concerns the mechanisms used to establish trust in dynamic networks of emerging cyber physical devices, such as smart phones and embedded computers in vehicles, that combine sensors, such as cameras, microphones and GPS, with general-purpose computing environments that offer significant communication and data storage capabilities.

The opportunistic nature of dynamic networks enables a rich set of distributed computations. However, it also exposes participating devices to a variety of security threats. A dynamic network may consist of several thousand devices of unknown provenance, and some may even be malicious in intent. A querying device must have mechanisms to establish the trustworthiness of other devices in the dynamic network in order to trust the results of its query. Similarly, a device that processes queries on behalf of other devices must have mechanisms to ensure the authenticity of devices issuing the query.

Trusted computing technologies offer a promising way to address the above problems. They enable a querying device to attest a remote device by establishing the integrity of the software stack executing on that device. In this paper, we focus on *hardware-based attestation*, which requires the remote device to be equipped with a

root of trust, such as a Trusted Platform Module (TPM) chip [11], that is used to bootstrap *integrity measurements* on that device. Attestation protocols (e.g., [6, 5]) transmit these integrity measurements (along with a description of the software stack of the remote device) to a querying device, which can then determine whether to trust the software executing on a remote device.

Unfortunately, the resource-constrained nature of cyber physical devices pose a key challenge that precludes direct application of previously-proposed trusted establishment techniques. Dynamic networks consist of mobile devices and sensors that are often constrained in battery power, network bandwidth and available CPU cycles. Existing attestation protocols require each participating device to transmit tens of kilobytes of integrity measurement data *each time* another device wishes to establish its trustworthiness. Because devices participating in dynamic networks may perform tens or even hundreds of computations each hour, directly deploying these protocols will place severe demands on resource-constrained devices.

We take the first steps to measure and understand the resource bottlenecks of hardware-based attestation in dynamic networks of resource-constrained devices. To that end, we use the **Sarana** macroprogramming architecture [4] for dynamic networks as our prototype architecture, and measure the impact of adding hardware-based trust establishment to this architecture.

**Prior work and contributions.** There is much recent attention on the topic of trustworthy mobile sensing [2, 8, 3, 9] to provide consumers with the assurance that the output of the sensors has not been tampered by malicious code. Much of this work has focused on the mechanisms to establish trust, such as introducing trusted hardware within the sensors themselves [9] to virtualized architectures for trusted computing [3]. Our work differs from these prior works in two respects. Foremost, to our knowledge, we are the first to study the energy considerations of incorporating trust-establishment in mobile sensing. Second, prior work has largely focused on the mechanisms and protocols for individual devices, while we are the first to consider the overall effects of trust establishment for dynamic networks of devices.

## 2 Background

We describe **dynamic networks** using an "Amber Alert" application, whose goal is to notify a population about emergency situations, such as a kidnapped child. Figure 1 illustrates how Amber Alert is implemented in the Sarana infrastructure. This figure shows a search application (e.g., "find all red four door sedans in the downtown area") that a *launcher* device would issue. In turn, the computation is automatically distributed by the Sarana runtime to participating *launchee* devices, such as cameras and cellular phones, within a geographic region (in lines 5 and 6). The launchees may run computations, such as image-processing software, on behalf of the launchers to find images that match the query (line 13). Launchees may themselves distribute this computation to other devices.

The Amber Alert application illustrates several features of dynamic networks. First, dynamic networks are loose confederations of mobile devices and other wireless sensors. They are characterized by the opportunistic nature of confederations, e.g., devices may choose to enter and leave the Amber Alert application at any time. Second, devices participating in dynamic networks share their resources to perform in-network distributed computations (e.g., image-processing in the Amber Alert application) on

```
1. public static void main(String[] args)
2.   SearchAttributes searchAttr = new SearchAttributes();
3.   searchAttr.parse(args); ......// Record attributes of search target e.g., "red
four-door sedan"
4.   Container carInfo = new Container();
5.   spatialview sv1 = camera @ Target_Geographic_Location
6.   visiteach cam ∈ sv1 every 30 secs within 3 hours
7.     boolean successfulMatch = false;
8.     Image img = cam.getImage();
9.     Time time = System.currentTime();
10.     Location loc = System.currentLocation();
11.     spatialview sv2 = imageUnderstandingCode @ new Circle(loc, 200m);
12.     visitone imageAnalysis ∈ sv2
13.       if (imageAnalysis.processImage(searchAttr, img) = match)
14.         successfulMatch = true;
15.     if (successfulMatch)
16.       carInfo.addElement(loc, time, img);
17.       spatialview sv3 = AmberAlertDisplay @ new Circle(loc, 100m);
18.       visiteach participant ∈ sv3
19.         participant.display(loc, time, img)
20.   carInfo.displayAll(); ........................// Display all images on launcher
```

**Fig. 1.** Sarana query example: Basic Amber Alert application.

behalf of other devices. This cloud-like ability to perform computations distinguishes dynamic networks from related work on ad hoc sensor networks, where the main goal is to gather and report data. Third, dynamic networks are especially well suited to efficiently process data obtained from the physical world. The opportunistic nature of dynamic networks allows a launcher device to specify exactly those devices that must perform a computation. For example, in the Amber Alert application, a launcher can restrict query processing to devices located within a geographic region.

The Amber Alert application also illustrates key security challenges in deploying dynamic networks. Devices in a dynamic network are untrusted and may misbehave, either out of malice or faulty software. For example, a launchee device may maliciously report false query results or overcharge a launcher for performing computation. It is therefore necessary for a launcher device to be able to establish the integrity of the software stack executing on a launchee device—this process is called *attestation* [7, 10]. Furthermore, because a launchee device may itself distribute the computation to other launchees, attestation must be transitive, i.e., the launcher must be convinced that all the devices participating in the computation are trustworthy.

**Hardware-based attestation** techniques offer one way to address some of the challenges of establishing trust in dynamic networks. Using attestation, a *verifier*, such as a launcher device, can determine the integrity of the software stack executing on a *prover*, such as a launchee device. To achieve this goal, hardware-based attestation techniques require that the prover have a hardware root of trust, such as a TPM chip. Such trusted hardware can be leveraged to gather, store, and cryptographically tie integrity assertions to the physical hardware as early as during the BIOS boot process.

Attestation protocols leverage the features provided by trusted hardware. The TPM chip has a set of internal registers (called PCRs), whose values are protected from device hardware and software. The value $v$ stored in a PCR register can only be *extended* using a new value $h$ using a `TPM_extend` operation. This operation updates the value in the PCR register to SHA-1($v||h$). Attestation protocols ([6, 5]) leverage this feature to construct a secure log of the code that is loaded for execution on the device. For example, the BIOS is modified to extend the value in the PCR with a SHA-1 hash of the bootloader. Similarly, the bootloader is modified to extend the PCR with a SHA-1 hash of the code of the operating system. The PCR thus contains an aggregate *measurement*

of the code loaded on the device. The device also maintains (separately, on its file system) a *measurement log* that stores the order in which code was loaded (e.g., the order in which modules were loaded or applications were initialized).

Each TPM chip is also associated with a public/private key pair that is used during attestation. A verifier can offer a challenge to a prover and request a fresh *quote* of the prover's state. This quote is a digital signature of the PCR contents and the received challenge. The prover sends the digital signature along with the measurement log to the prover. Using the log, the verifier can recreate the aggregate measurement in the PCR and assess the prover's integrity based on the integrity measurements.

Existing attestation techniques have two key shortcomings that make them inapplicable for use with resource-constrained cyber-physical devices. First, most attestation protocols transfer large amounts of data between the prover and verifier. For example, about 1000 code measurements are typical for a commodity Linux installation [6]. We do not expect this number to change significantly even on cyber physical devices. Transferring this data (in the form of measurement logs) between devices will likely place severe energy demands on cyber physical devices.

Second, attestation involves a challenge/response protocol between a prover and verifier. The protocol must be executed *each time* a prover attempts to establish its integrity to a verifier. Coupled with large data transfers for each attestation attempt, the interactive nature of this protocol makes it impractical for use with dynamic networks, where each launchee (prover) may perform several computations each hour on behalf of multiple launchers (verifiers). Furthermore, verifiers in dynamic networks may themselves be resource-constrained. Because attestation protocols require significant computation and storage even on verifiers, e.g., to execute the protocol, perform cryptographic operations to validate the prover and to collect and store acceptable code measurements, these protocols are resource intensive even for verifiers.

## 3 Model

We now discuss the execution model of the Sarana system both with and without attestation. This model is the basis for our experimental study. In Sarana, the *visit* statement is the language construct that allows programmers to specify in network computation. When executing a visit statement on a launcher device, the body of the visit statement, i.e., its iterations are sent to different network nodes (launchees) for execution. Launchees are selected based on their location and services that they provide. For example, the Amber Alert program shown in Figure 1 specifies visits across dynamic networks of cameras, displays, and image analysis nodes.

A Sarana program can be as simple as a single visit statement that collects some basic information for each visited node, and reports back some summary information. A sketch of this program is shown in Figure 2(a). Figure 2(b) is a simplified sketch of our Amber Alert program shown in Figure 1. Both program sketches will be used in our experiments.

Figure 3 provides an overview of the the basic execution model for a single visit statement. Consider the left side of Figure 3, which executes without attestation. The launcher (injection node) sends code and data needed to execute a single parallel iteration to each dyanmic network node (launchee). The launchees receive the code and data, execute it, and send the results back to the launcher. Finally, the launcher receives

```
1. spatialregion cs = Service;
2. sum_reduction int count = 0;
3. visit c in cs { count += 1; }
4. system.print(count);
```

**(a) Single visit example.**

```
 1. spatialregion cs = CameraService;
 2. visit c in cs {
 3.     Image image = c.takePhoto();
 4.     spatialregion as = AnalysisService;
 5.     or_reduction boolean test = false;
 6.     visitone a in as {
 7.         test |= a.analyze(image);
 8.     }
 9.     if(test) {
10.         spatialregion ds = DisplayService;
11.         visit d in ds { display.show(image); }
12.     }
13. }
```

**(b) Nested visit example.**

**Fig. 2.** Examples of Sarana programs.

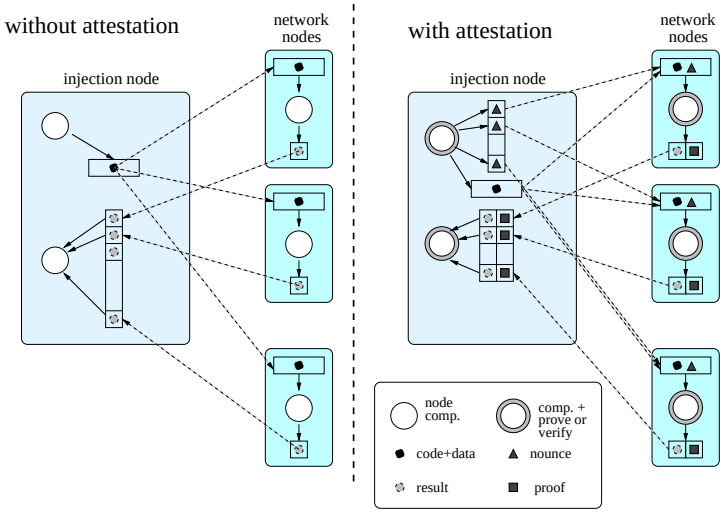


**Fig. 3.** Graphical representation of execution model of a single visit statement.

the results and aggregates the results (e.g., in Figure2(a) the aggregation operation is summation).

Our proposed attestation model, as shown in the right side of Figure 3, is similar. Here, before sending the code and the data to the launchee devices, the launcher computes a challenge (which is a random number, called a *nonce*) sequentially and sends it to the launchees. The launchees, must prove the integrity of their software stack to the launcher. So each launchee device sends one message to the launcher, which includes the measurement list, the aggregate value of the list, the signed nonce and the result of the computation. Finally, before aggregating the results of the visit iterations, the launcher device verifies the attestation data sent by the launchee devices and filters the results. The same execution model applies even in the case of *nested visits*, where a launchee device can itself delegate computation to other nodes in the dynamic network.

# 4 Evaluation

We implemented the Sarana dynamic network system and the attestation protocol described earlier in a discrete event-based simulator. This simulator explicitly models time utilization and measures energy based on a workload's usage of resources. To model the time consumption of TPM operations, our simulator uses measurements obtained from a TPM-enabled desktop machine. We used the TrouSerS library [1] to measure the time needed for a prover (to produce and send a measurement list) and a verifier (to verify a digital signature). Our measurements indicate that a prover uses 732 milliseconds and the verifier, including reading from the NIC, uses 6,686 microseconds. For all other timing models (e.g., the time taken to aggregate results from launchees), we use measurements obtained by profiling a Nokia N900.

## 4.1 Applications

We use two representative applications to evaluate our work. The first application consists of only one visit statement, which is the most common control flow graph for programs sampling the surrounding environment. The second application's control flow graph is more complicated, and also representative of how nested **visit** statements interact.

Due to the generic behavior of our first application, we will evaluate with a number of timings for the visit body, which we refer to as payloads. The source code in Figure 2(a) shows a single addition operation, but this program could be reading a sensor or a more complex computation. As such, our evaluations will use the payloads 0 microseconds, 500 microseconds, 1 milliseconds and 500 milliseconds. The payload for Amber Alert needs to represent the computation realistically. From our experience, an automated camera operation is 200 milliseconds, image analysis is 500 milliseconds and displaying an image is at least 1 second. Our version from Amber Alert differs from the original Sarana project's Amber Alert. No human interaction controls the camera, but a user can control display to save the image to the launchee's device for later investigation. Our motivation is to minimize the resource utilization for participating devices and their owners.

Another dimension to explore is the number of nodes in the network. Notice in Figure 2(b) that each visit statement loops over a *spatialregion* which itself is defined with respect to specific services that a node may or may not offer the system. Our Amber Alert evaluations assume a fixed proportion of the services, which we define to be 33% cameras, 20% image analyzers and 47% displays. Since our other application does not specify any services, this is not a factor in its evaluation.

By nature, applications executing over a dynamic network can be very difficult to understand. Our results in this paper purposefully do not take in account temporal or spatial concerns with respect to the applications semantics.

**Application Configurations:** We use various configurations of the applications in Figure 2 to evaluate the energy consumption of two representative tasks in a dynamic network. Each configuration of the first application varies the amount of time spent in the *visit* body. We simulate configurations in which the time spent visiting other nodes is $0\mu$s, $500\mu$s, 1ms and 500ms. For the second application, which is a simplified version of Amber Alert, our simulator assumes that the automated camera operation consumes 200ms, image analysis takes 500ms, and that displaying an image takes at least 1 second; we obtained these numbers from a prior deployment of Sarana [4]. We also assume

that in our dynamic network, 33% of the nodes are cameras, 20% are image analyzers and the remaining 47% are displays. These parameters determine the frequency with which various branches of the code in Figure 2(b) are taken.

### 4.2 Simulation Results

**Timing.** Figures 4 and 5 show the results of simulating the code in Figures 2(a) and (b). Figure 4 shows that with attestation, the overall time to execute the code in Figure 2(a) grows approximately linearly with the number of nodes (for various durations of the *visit* statement). The time accounted for attestation is less than a minute even for networks under 1,000 devices. In Figure 5, the timing with attestation is only slightly slower than the timing without. In absolute terms, the overhead of attestation is about 10 seconds until the network reaches a size of 1000 nodes, following which the overhead increases at a super-linear rate.

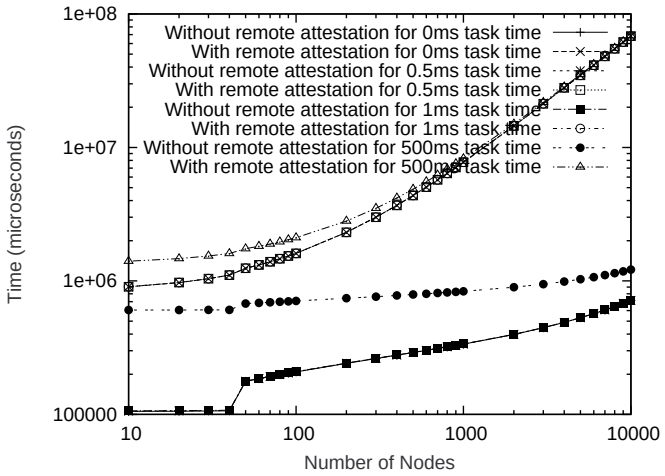


**Fig. 4.** Simulation results for Figure 2(a).

**Energy.** The energy consumption of the code in Figure 2(a) for various configurations is shown in Figure 6. For visit times that range between 0$\mu$s and 1ms, the percentage of energy consumed by attestation is 97% or greater, a very significant fraction of the overall energy budget. For the configuration with 500ms visit times, the energy consumption of attestation drops, but is still a significant 58% of the overall energy budget.

Notice that a verification portion of the attestation work is performed by the launcher device. This means that the launcher device is utilizing a large portion of the energy that the entire network is utilizing for this application, which implies that the launchers of large visit loops cannot be energy-constrained. In fact, the single visit 0$\mu$s payload version is a worst-case scenario with respect to attestation energy utilization. The size of the network does not impact this benchmark. The events of the computation are proportional with the size of the network, and so the results are less than 1% percent different when comparing networks of size 10 to 10,000.

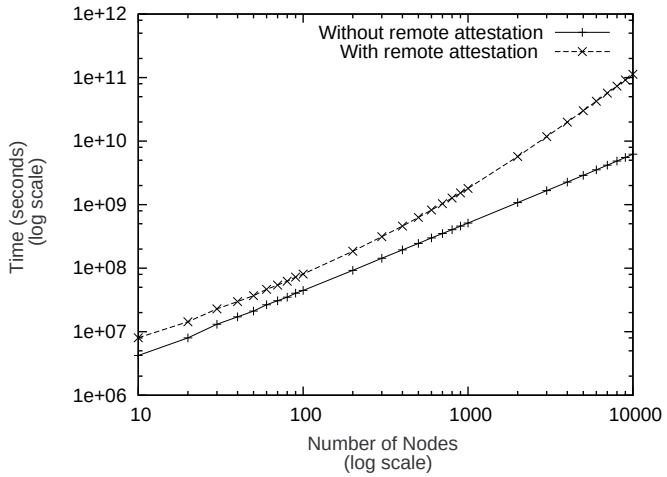


**Fig. 5.** Simulation results for Figure 2(b).

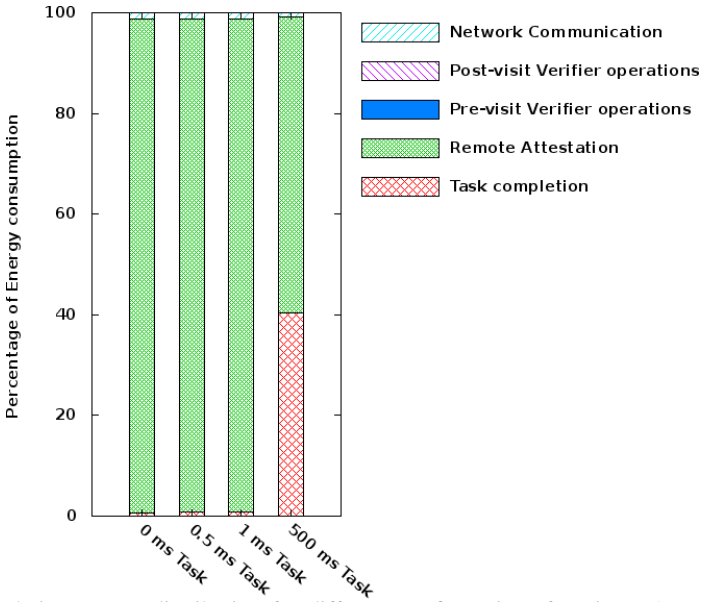


**Fig. 6.** Simulation energy distribution for different configurations for Figure 2(a). X-axis is 0 seconds, 0.5ms, 1ms and 500ms. Y-axis is energy consumption as a percentage. The items inside the bars are network communication, pre-visit operations, post-visit operations, attestation and payload.

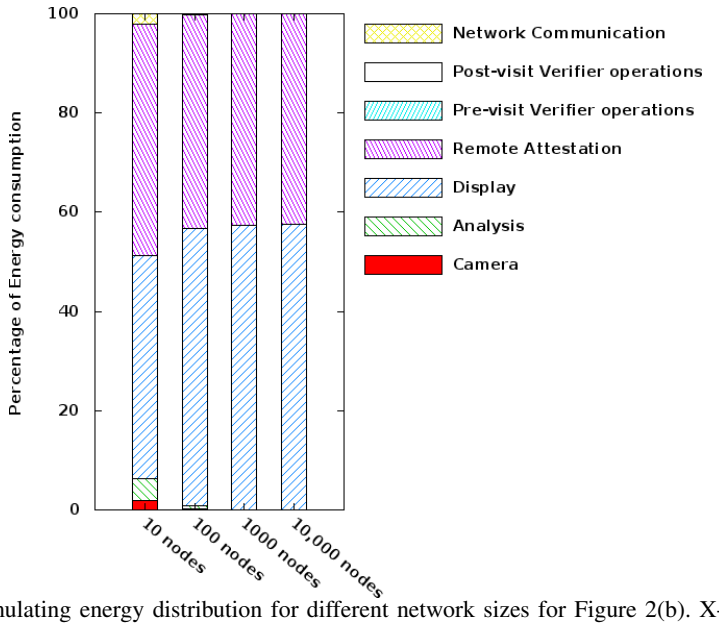


**Fig. 7.** Simulating energy distribution for different network sizes for Figure 2(b). X-axis is 10 nodes, 100 nodes, 1000 nodes and 10,000 nodes. Y-axis is energy consumption as a percentage. The items inside the bars are network communication, pre-visit operations, post-visit operations, attestation, display, analysis and camera.

The energy consumption of the code in Figure 2(b) depends on the ratio of services (i.e., cameras, image analyzers and displays) in the network. We expect that as the size of the network grows, the percentage of energy for taking photos and analyzing photos to be linear, while the percentage of energy for displays images to grow quadratically with respect to the energy just for visit loop bodies (i.e., excluding network, attestation, etc.). The reason is that for a network with $n$ nodes, there are $n/3$ visit iterations for the camera services of the network and for the image analysis, and there are $7n^2/45$ total visit iterations for the display services. However there is also a significant cost incurred by attestation. In fact, the number of verifier-prover calls is quadratic, $(7n^2 + 30n)/45$, with respect to the size of the network.

Figure 7 shows the energy results for a network of size 10. Attestation consumes 46% of the total energy requirements for this benchmark. The energy consumption does change as the network sizes grows, for instance with 10,000 devices 42% of the energy is for attestation. Essentially, attestation doubles the energy requirement for a Sarana application like Amber Alert.

**Discussion.** Applications such as the ones in Figure 2 spend a large fraction of their overall energy budget on attestation. While it is a matter of preference as to what limit is too much, doubling or more of the total energy budget may be too high for many.

We believe that while this is very large for the above applications, there are applications for which attestation would be much smaller percentage of the energy budget. For instance if the payload is minutes long, attestation is much less noticeable. Examples of greater payload applications would be media sharing and collaborative document creation. While that decreases the overall percentage of attestation energy usage, it does not solve the problem of the energy-constrained launcher. We propose that a

tree-based propagation of visit iterations would decrease all launchers energy usage, and also can decrease the total time for the application to complete. These solutions can make energy-constrained attestation feasible.

## 5 Conclusion and Future Work

We measured the impact of hardware-based attestation on resource utilization in the Sarana dynamic network system. Our evaluation using timings shows that a system with attestation scales just as well as a system without attestation. However, we find that the overall energy drain in an attested network can be significant: nearly doubling the overall energy usage of an application in the dynamic network.

Given the increasing importance of cyber physical devices, and the utility of connecting them opportunistically using dynamic networks, we believe that the problem of establishing trust in such networks is paramount. The study in this paper shows that energy can be a significant barrier to the adoption of existing attestation protocols in such dynamic networks, particularly for small node payloads. We therefore conclude with a call for future research on energy-efficient trusted computing protocols, tailored for dynamic networks of cyber physical devices.

## References

1. Trousers. http://trousers.sourceforge.net.
2. A. Dua, N. Bulusu, W. Feng, and W. Hu. Towards trustworthy participatory sensing. In *USENIX Workshop on Hot Topics in Security*, 2009.
3. P. Gilbert, L. Cox, J. Jung, and D. Wetherall. Toward trustworthy mobile sensing. In *11th Workshop on Mobile Computing Systems and Applications*, 2010.
4. P. Hari, K. Ko, E. Koukoumidis, U. Kremer, M. Martonosi, D. Ottoni, L-S. Peh, and P. Zhang. SARANA: Language, compiler, and runtime system support for spatially-aware and resource-aware mobile computing. *Philosophical Transactions of the Royal Society A*, 366(1881), October 2008.
5. T. Jaeger, R. Sailer, and U. Shankar. PRIMA: Policy-Reduced Integrity Measurement Architecture. In *Proceedings of the 11th ACM Symposium on Access Control Models and Technologies (SACMAT)*, June 2006.
6. R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, USA, August 2004.
7. Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *USENIX Security*, 2004.
8. S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *10th Workshop on Mobile Computing Systems and Applications*, 2009.
9. S. Saroiu and A. Wolman. I am a sensor, and I approve this message. In *11th Workshop on Mobile Computing Systems and Applications*, 2010.
10. TCG. Trusted Computing Group: Glossary of Terms. https://www.trustedcomputinggroup.org/groups/glossary/.
11. TCG. Trusted Computing Group: Trusted Platform Module (TPM) Specifications. https://www.trustedcomputinggroup.org/specs/TPM/.