Combating Software and Sybil Attacks to Data Integrity in Crowd-Sourced Embedded Systems

AKSHAY DUA, NIRUPAMA BULUSU, and WU-CHANG FENG, Portland State University
WEN HU, Commonwealth Scientific and Industrial Research Organisation, Australia

Crowd-sourced mobile embedded systems allow people to contribute sensor data, for critical applications, including transportation, emergency response and eHealth. Data integrity becomes imperative as malicious participants can launch software and Sybil attacks modifying the sensing platform and data. To address these attacks, we develop (1) a Trusted Sensing Peripheral (TSP) enabling collection of high-integrity raw or aggregated data, and participation in applications requiring additional modalities; and (2) a Secure Tasking and Aggregation Protocol (STAP) enabling aggregation of TSP trusted readings by untrusted intermediaries, while efficiently detecting fabricators. Evaluations demonstrate that TSP and STAP are practical and energy-efficient.

Categories and Subject Descriptors: C.2 [**Computer Systems Organization**]; C.3 [**Special-purpose and Application-Based Systems**]: *Real-time and embedded systems*; C.4 [**Performance of Systems**]: *Design studies*

General Terms: Algorithms, Design, Experimentation, Measurement, Performance, Reliability

Additional Key Words and Phrases: Trust, security, mobile computing, embedded systems, crowd-sourced sensing, critical systems, data integrity

ACM Reference Format:
Akshay Dua, Nirupama Bulusu, Wu-Chang Feng, and Wen Hu. 2014. Combating software and sybil attacks to data integrity in crowd-sourced embedded systems. ACM Trans. Embedd. Comput. Syst. 13, 5s, Article 153 (August 2014), 19 pages.
DOI: http://dx.doi.org/10.1145/2629338

1. INTRODUCTION

Today, there are over four billion mobile phone users in the world, many of whom use smartphones. These embedded systems are equipped with built-in positioning, audio, video, and other sensors and could potentially interface with other sensor-equipped devices (e.g. automotives and pacemakers). Crowd-sourced sensing from mobile embedded systems offers an unprecedented opportunity to connect our world for critical applications ranging from intelligent transportation [Hull et al. 2006], disaster response, urban monitoring [Agapie et al. 2008], public safety, public health [Reddy et al. 2007], and real-time citizen journalism [CNN] through mobile social applications and

This research was supported by the National Science Foundation under NSF CAREER award 0747442.
A preliminary version of this article appeared in the Usenix Workshop on Hot Topics in Security (HotSec'09) [Dua et al. 2009]. This article features significant new measurements on the trusted sensing peripheral, and new contributions on the design and evaluation of a secure and trusted data aggregation protocol, as well as extending the trusted sensing peripheral to legacy mobile devices.
Authors' addresses: A. Dua, N. Bulusu (corresponding author), and W.-C. Feng, Portland State University, Department of Computer Science, 1900 SW 4th Ave, Suite 120, Portland, OR 97201. phone: (503) 725 4036; fax: (508) 725 3211; email: {akshay, nbulusu, wuchang}@cs.pdx.edu; W. Hu: Commonwealth Scientific and Industrial Research Organization, 1 Technology Court, Pullenvale, QLD 4069, Australia; phone: 61 7 3327 4043; fax: 61 7 3327 4455; email: wen.hu@csiro.au.

DOI: http://dx.doi.org/10.1145/2629338

instruments. All that is required then, is for sensor-equipped mobile devices to publish readings to Internet portals like SensorMap [Nath et al. 2006] while people go about their everyday lives.

This use of existing mobile communications infrastructure makes crowd-sourced mobile embedded sensing systems cheaper to deploy than dedicated wireless sensor networks. Data collected from such systems can improve understanding of local environments, help shape public policy, facilitate new scientific research, or facilitate large-scale clinical studies. However, government and researchers alike will be reluctant to use the data if they cannot trust its integrity.

This lack of trust would be unfounded if sensed data were infeasible to fabricate, but that is not the case. The openness of the Internet infrastructure and the potentially critical role of data provide both a massive opportunity and an incentive to tamper with these data flows. Consider a crowd-sourced traffic navigation system like Waze [Waze]. Participants of Waze download a client application on their smartphones that activates the phone's internal GPS receiver, and periodically reports their GPS coordinates to Waze. With sufficient user participation, Waze can provide optimal routes to destinations based on the real-time traffic information learned from participants. A malicious participant could easily fool Waze by modifying the client application to change the GPS coordinates or delay publishing them. He could launch multiple emulated smartphones on resource-abundant hardware and submit random GPS coordinates. We classify the former as a *software attack* and the latter as a *Sybil attack* [Douceur 2002]. Such data distortion attacks could be motivated by mischief, personal gain, or malice. For example, a user may want to make a neighborhood street appear congested when it is not, so others do not take that route. Although our example considers GPS, the software and Sybil attacks are applicable to any of the phone's sensors.

Moreover, a common thread across crowd-sourced sensing applications is in shaping data from communication-equipped mobile embedded systems into tools for understanding the complexity of our physical world and responding to it swiftly, often working without human intervention. To achieve this transformation, the massive volumes of data originating from the sensors, actuators, and other devices and flowing into back-end computers must be automatically aggregated and analyzed for rapid decision-making. In any sophisticated system, such rapid analysis leverages aggregated data for efficiency; rather than relying on raw data. However, aggregation is also vulnerable to data fabrication attacks, as an aggregator can easily pollute the aggregate value.

This article presents a trusted hardware-based sensing platform to combat these attacks, enabling crowd-sourced high-integrity data collection, of either raw or aggregate data. The contributions of this article are the following.

—*The design and implementation of a Trusted Sensing Peripheral* (TSP). The TSP is a sensing platform that consists of a Trusted Platform Module (TPM) [Trusted Computing Group c] and hardware sensors. Our goal is to make the TSP economically infeasible to emulate or modify without being detected. The TPM facilitates this by attesting to the authenticity and integrity of the TSP as well as the sensor data. Successfully verified attestations prove to a data receiver, such as an online portal, that the data is indeed authentic and was not altered, either by the TSP or during transit. Experiments show that the TSP is very energy efficient: it has a projected battery life of over 80 days when sampling, attesting, and transmitting a temperature sample every 30 seconds.

—*An efficient, end-to-end, high-integrity, data aggregation protocol for crowd-sourced mobile embedded sensing systems.* The TSP publishes sensor data to an online portal via the participant's mobile device acting as a forwarding proxy. To reduce the energy expended in transmitting data and the processing overhead at the portal, the mobile

proxy has the option of aggregating the TSP's raw data before forwarding it. However, the online portal has no way of trusting that the aggregation was performed correctly. To provide aggregation integrity guaratees to the portal, we developed STAP (Secure Tasking and Data Aggregation Protocol), a protocol modeled on a bit-commitment scheme [Chaum et al. 1987; Baughman and Levine 2001]. STAP uses a pseudo-random challenge-response mechanism that allows a portal to detect a lying proxy. Our results indicate that by checking only 20% of the aggregates, the portal can detect a lying proxy within the first six false aggregates received.

—*A high-integrity platform that augments legacy mobile devices with trustworthy sensing.* The Trusted Sensing Peripheral (TSP) enables legacy untrustworthy mobile devices to produce trusted data. Section 6.2 discusses challenges to making contemporary sensing-capable phones trustworthy: namely the absence of proper hardware support, distrust in trusted hardware, and growing exposure to remote attacks. Besides trustworthy sensing, the TSP provides greater sensing modalities than commodity mobile devices.

2. RELATED WORK

Existing research on crowd-sourced mobile embedded sensing systems does not address the data integrity problem the way we define it, namely, "How can a data portal—receiving data from sensors not under its control—trust that the data is a true representation of the real-world phenomenon being sensed?".

Research on traditional sensing, however, suggests some approaches that might be adapted to solving the problem. For example, a reputation-based framework implemented at the portal could identify and ignore data from sources with a low reputation [Ganeriwal et al. 2008]. A source is assigned a low reputation as long as it generates outlying data samples when compared to others in its neighborhood. Note that this approach does not provide protection from Sybil attacks, where an adversary could create any number of virtual sources and use them to artificially raise a given source's reputation. A reputation-based system will also be more effective in detecting faulty sources than malicious ones, because malicious sources will try to avoid detection by publishing fabricated, but not necessarily outlying data [Ganeriwal et al. 2008]. Among other issues, reputation-based frameworks require redundant sources of data to detect outliers, and are specific to the type of data being collected.

Another approach, also from traditional sensing, involves filtering fabricated data that is injected into the sensor network [Zhu et al. 2004]. Data that at least $t + 1$ local sensors don't agree on, or data coming from unauthentic sensors is considered fabricated. The system is thus resilient to adversaries that have compromised at most t sensor nodes. Since each sensor in the network can be authenticated, the sensor network is resilient to Sybil attacks. However, the assumption is that each sensor node already shares a key with the authenticator (the base station). This assumption might be realistic for traditional wireless networked embedded systems, but is not realistic for crowd-sourced mobile embedded sensing systems, where the data portals and data producers (participants from the crowd) are different autonomous entities. It would be unreasonable to expect that every potential data portal has a preexisting secret key with every potential data producer. The Sybil attack, could thus be launched by a data producer during the time a shared secret key is established with a new data portal. Another disadvantage is that software attacks where data is modified after being sensed, can be detected only when less than $t + 1$ nodes have been compromised.

An orthogonal approach, as opposed to these, and closer to what is presented in this article, is for data portals to use systems like Pioneer or SWATT (SoftWare-based ATTestation) to externally verify the code executing on a remote data producer's platform [Seshadri et al. 2004, 2005]. With assurance in the producer's sensing platform, the

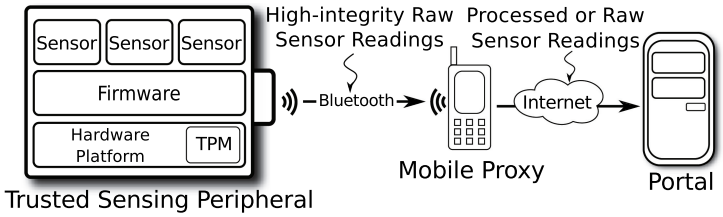


Fig. 1. The TSP publishes data via the participant's mobile device, which can then process the data, or forward it as is.

portal will be inclined to trust data from it. Unlike our trusted-hardware approach, both SWATT and Pioneer are software-based systems that challenge the remote platform to compute a digest of its memory contents. The digest is computed by traversing the memory in a pseudo-random fashion as determined by the random challenge. The challenger is expected to know the memory contents of the remote platform beforehand and can therefore independently compute the correct digest for verification later in the protocol. A remote platform could fool the challenger by separately storing the original contents of the modified portions of memory, allowing it to still provide the correct digest. However, since the traversal is pseudo-random, the malicious platform cannot know beforehand the order in which the memory will be checked. As a result, the attacker will have to check each memory access to see if the address matches one of the modified ones. This extra check will cause the digest computation operation to take longer than expected, causing the challenger to become suspicious. Notice though, that neither SWATT not Pioneer can prevent a Sybil attack from an adversary that has abundant computational resources. The attacker could simulate the remote platform on more powerful hardware while still meeting the expected response time. Since the response time of the remote platform is an estimate based on the hardware configuration and the expected communication latency, SWATT and Pioneer only provide a probabilistic guarantee of the remote platform's integrity. Estimating response times may itself be a daunting task.

More recently, YouProve [Gilbert et al. 2011] studied the question of how to verify the authenticity of user-modified data. Their focus is on audio and images, rather than aggregated scalar data. Like our earlier work, their approach explores how a Trusted Platform Module could be leveraged. Their work assumes the eventual availability of a TPM on a smartphone. Unlike us, they do not actually implement their system using a physical TPM due to the nonavailability of TPMs on current commodity smartphones.

Our protocol is inspired by an approach in traditional sensing called Secure Information Aggregation (SIA) [Przydatek et al. 2003]. Most research on fault-tolerant sensor data aggregation focuses on faults that arise due to the natural data loss arising from the noise and asynchrony of physical deployments [Denantes et al. 2008; Sharma et al. 2007], rather than due to a malicious adversary. SIA, however, assumes a malicious adversary. However, unlike our approach, SIA assumes that the sensors are trustworthy and does not provide an implementation of the protocols. Our framework makes it easy to incorporate other SIA algorithms that are more suitable for the respective aggregation function.

3. SYSTEM MODEL AND TRUST RELATIONSHIPS

As Figure 1 shows, the TSP collects sensor data, has the TPM sign it, and forwards it to the mobile proxy, which either aggregates the raw data or forwards it as is. The portal does not trust that the TSP is functioning as expected, and thus requires that the TSP prove its trustworthiness. The portal also does not trust the mobile proxy to correctly perform the aggregation and expects proof of aggregation integrity. An

aggregation may be incorrectly performed by the mobile proxy if it does not use inputs from the TSP, or if it uses a different aggregation function than what is expected by the portal. In either case, once the TSP proves that it is indeed trustworthy, it assists the portal with establishing aggregation integrity. The portal is assumed to be trusted by all parties. However, it can prove its true identity by presenting a public-key certificate signed by a trusted authority. The certificate can be presented using standard secure communication protocols like SSL/TLS [openssl.org].

4. THREAT MODEL

In our system, we consider threats to the proper operation of the TSP and the mobile proxy.

4.1. Threats to the TSP

Our goal is to prevent or detect software and Sybil attacks [Douceur 2002] against the TSP. This will ensure that data reported to the portal was not modified in unintended ways by the TSP software, and did not originate from emulated, simulated, or fake devices. We considered the following specific threats.

—*Software Attacks.* This refers to any modification of the TSP firmware or of the memory at run-time. Such modifications can be used to compromise the integrity of the data collected by the sensor. Protecting the TSP from software modification attacks ensures that the data output is the same as the data collected by the sensors on the TSP. Thus, preventing software modification also preserves data integrity.
—*Sybil Attacks.* The adversary can have a device that either pretends to be a legitimate TSP, or creates multiple emulated or simulated TSPs. Each instance of such a fake TSP is called a Sybil [Douceur 2002]. The Sybil attack can only be successful if the system cannot reliably distinguish between a real and a fake TSP.

4.2. Threats to the Mobile Proxy

The participant's mobile device serves either as a forwarding proxy for raw data originating from the TSP, or as a data aggregating proxy when processing the raw data before sending it to the portal. A malicious participant may modify its mobile proxy to compromise the integrity of published data. Thus, the proxy is not trusted by the portal or the TSP and is assumed to pose the malicious data aggregation threat. Malicious data aggregation involves a compromised proxy faking the aggregate values, dropping them, or injecting any new ones. In this scenario, the portal must be able to detect and reject data from the malicious proxy.

4.3. Threats to Communication

A man-in-the-middle can inject, modify, drop, or replay messages in transit between the TSP and the portal. This includes any changes made by the mobile proxy to the data it is forwarding, or any changes made to the data being transmitted between the TSP and the proxy, or between the proxy and the portal. A successful attack on communication integrity can result in the publication of false, corrupt, or stale data. Moreover, these threats are the same as those posed by a compromised forwarding proxy. Thus, addressing communication integrity threats simultaneously addresses threats from a malicious proxy.

4.4. Threats not Addressed

Although we address some of the most challenging threats to data integrity in crowd-sourced mobile embedded sensing systems, the following threats have not been addressed.

—*Availability.* A remote adversary may force the TSP or proxy to continuously receive messages, depleting their battery [Stajano and Anderson 2000]. Or the mobile proxy may drop all communications between the TSP and the portal. Such forms of Denial of Service (DoS) attacks are currently not addressed.
—*Confidentiality.* We primarily address data integrity in crowd-sourced mobile embedded sensing systems as such, we rely on other components to provide confidentiality. For example, the channel between the TSP and the mobile proxy (see Figure 1) can be secured using Bluetooth's security features [Bluetooth Special Interest Group 2009; Padgette et al. 2012].
—*Physical Data Poisoning.* A participating adversary may alter the very phenomenon being sensed. For example, he could collude with others to drive slowly, thus, depicting congestion. It is important to note that the TSP necessitates physical tampering of the phenomenon to publish fabricated data, as opposed to simple software modification to do the same. We believe this greatly decreases the likelihood of data-poisoning attacks.
—*Hardware attacks on the TPM.* Since the TPM is the root of trust for our approach, we assume that it is tamper-proof as claimed, not compromised, and functioning as expected. This is a reasonable assumption given that simple yet effective hardware attacks on the TPM, like the timing and reset attacks, have already been addressed [Atmel Corporation; Trusted Computing Group b]. Furthermore, the most recent attack involving extraction of an obsolete TPM's burned-in private key took specialized skills, around six months, and costly equipment worth about 200,000 USD [The H. Security 2010]!
—*Privacy.* The TSP is a sensing device that is carried by paraticipants as they go about their daily lives. Sensitive data like location information published by the TSP could compromise the privacy of the participant. As a result, the participant must trust the portal with their sensitive data. Although mix networks like Tor [Dingledine et al. 2004] anonymize the origin of data, privacy threats due to sensitive information (e.g. location) in the data itself still remain. Moreover, protecting participant privacy while guaranteeing the integrity of the data they publish is a topic of active research and out of the scope of this article [Shi et al. 2011; Rastogi and Nath 2010; Popa et al. 2009]. We do, however, acknowledge the fundamental privacy implications of TPM on hardware and software configurations, and believe that our solution works as an opt-in service. The decision to attest (upload a fidelity certificate, as in YouProve [Gilbert et al. 2011]) is an explicit choice that remains under the users control. We see our solution as an opt-in service for users who wish to prove the authenticity of the data they contribute. Moreover, we believe that the importance of privacy itself and how it impacts user participation differs across users. The University of Cambridge researchers recently launched EmotionSense [Lathia et al. 2013], which logged sensitive information like location, acceleration, call time, call duration and so on, and streamed this data to a remote database. Many would think that privacy concerns might stifle the application, but EmotionSense currently has 10,000 active users and by using opt-in terms of service, has also passed the necessary ethical hurdles.

5. THE TRUSTED PLATFORM MODULE

As we discuss in Section 6, the TPM is the root of trust for data authenticity and platform integrity assurance provided by the TSP. The TPM is a tamper-proof secure coprocessor that enables trusted computing principles on commodity computing platforms. It is housed on the host platform and provides tamper-proof storage for cryptographic keying material, and platform configuration information. Additionally, it can digitally sign and report the securely stored configuration information, thus

indirectly attesting to the integrity of the platform. Further, since the digital signatures are computed using keys protected from extraction, any signed information from the TPM can be considered authentic once the signature is verified.

A TPM is associated with several credentials, each containing information regarding the TPM or its associated platform, and digitally signed by the entity issuing the credentials. References to the various TPM credentials (in the form of message digests) along with the public portion of the TPM's RSA signing key (AIK_{pub}: Attestation Identity Key) are included in a final *attestation identity credential* that is then signed by a trusted certificate authority. Once presented to remote entities, the successful verification of the attestation identity credential proves that the specified platform indeed contains a certified TPM and that any digital signatures performed by that TPM (using AIK_{priv}) can be verified using the included public signing key. Since the corresponding private key is securely created, stored, and protected from extraction by the TPM itself, the TPM's signature cannot be repudiated.

The TPM also provides load-time platform integrity verification via its platform attestation feature. However, run-time changes to the platform cannot be directly captured by the TPM. Section 6.2 discusses the design of our Trusted Sensing Peripheral (TSP), which inherently resists run-time software attacks. Thus, trust in the run-time state along with trust in the initial platform configuration transitively induces trust in the load-time state of the platform, allowing the TSP to use the TPM solely as a signing authority.

6. DESIGN

In this section, we describe the Trusted Sensing Peripheral (TSP) and the Secure Tasking and Data Aggregation protocol (STAP).

6.1. Design Assumptions

We assume that the online portal has the TPM's attestation identity credentials, which contain its public Attestation Identity Key AIK_{pub}, used to verify the TPM's signature. For STAP, we assume a secure transmission channel (e.g. the Secure Sockets Library (SSL)) between the proxy and the portal, and that the portal has a strong pseudo-random number generator (our protocol's security depends on the assumption that it is intractable for a malicious proxy to reliably predict the sequence of random numbers generated by the portal).

6.2. Design Rationale

Designing a high-integrity sensing platform is not trivial. Especially, when the platform can potentially be in the physical possession of an adversary. This, by-definition, is the case with crowd-sourced mobile embedded sensing systems: participants carry mobile platforms with built-in sensors that upload data to an online portal. Although most participants may be honest, one cannot ignore that malicious participants may poison data integrity (see Section 4).

At the very least, the portal must be able to detect sensory data tampering, or the sensing platform must be able to prevent it entirely. Since participation in crowd-sourced sensing is voluntary, people may register at any time with private heterogeneous sensing devices. Thus, the system must be able to detect fake devices or prevent their participation.

Cryptographic techniques that detect data tampering across a communication channel (e.g. message digests) work only if the communicating parties have a vested interest in protecting the transmitted data. However, in crowd-sourced mobile embedded sensing systems, one of the parties, namely the data producer, could very well be the

adversary, making it very hard for the portal to discover if fabricated data was being protected in the first place.

The portal could compare multiple data values produced in the same region and reject the outliers. Although this method may be effective at detecting outliers, it may not be effective with an adversary who wants to avoid detection (see Section 2). Consider again, the example of an adversary that emulates multiple distinct mobile devices and publishes GPS data to make a quiet neighborhood street appear congested. This situation will hardly appear to the system as outlying activity. Furthermore, since the portal cannot distinguish between real and emulated platforms, this sort of software collusion attack using multiple Sybil (or fake) identities will go undetected. Other issues with this approach are the need for redundant data sources in any region, and different outlier detection mechanisms depending on the type of data being collected.

Consequently, our design takes the prevention approach. Here, the sensing platform itself prevents any modification to the sensed data. We use a trusted third-party housed on the platform to vouch for the integrity of all the software running on it. If the portal cannot verify the presence of a trusted third-party on the remote sensing platform, it can choose not to trust the data emanating from it. Since the presence of the trusted third-party cannot be cloned or faked, it is impossible for a sensing platform to appear trustworthy when its not. Our trusted third-party housed on the sensing platform is the TPM. It forms the root of trust for data and platform integrity assurances provided by the associated sensing platform.

Recall that the sensing platform in crowd-sourced mobile embedded sensing systems is usually the participant's mobile device, which is assumed to already have existing sensing capabilities (e.g. GPS, accelerometer, microphone). However, our approach was to build a separate "closed box" high-integrity sensing platform we now call the Trusted Sensing Peripheral (TSP). Several factors motivated this approach.

—As Garfinkel et al. [2003] stated in their work on a trusted virtualization platform called Terra. "The security benefits of starting from scratch on a closed box special-purpose platform can be significant." They believed, like us, that an opaque special-purpose platform can more easily protect the integrity of its contents.

—Providing closed box semantics on most commodity mobile devices is not possible due to the lack of required hardware support and effective protection of software from run-time attacks. Terra [Garfinkel et al. 2003] provides load-time application integrity guarantees using the TPM, and strong isolation by running it inside a virtual machine managed by a Trusted Virtual Machine Monitor. However, we are not aware of any existing commodity mobile devices equipped with TPM chips. Further, Terra does not protect software from run-time attacks by malicious device drivers that have access to the DMA controller. Another system, proposed by McCune et al. [2008], called Flicker, provides a secure execution environment for a portion of an application's logic. Besides the need for a TPM, this approach also requires additional hardware support (like the *SKINIT* instruction found on certain AMD processors [Advanced Micro Devices], or the *GETSEC* + *SENTER* instructions on certain Intel processors [Intel Corporation]) to set up the secure execution environment. Flicker provides protection from DMA attacks and strong isolation, by disabling all interrupts and debugging support. However, regardless of hardware support, certain practical limitations remain. Flicker requires that the OS and all associated tasks be suspended during the time the secure execution environment is active. Given the high overhead of the *SKINIT* instruction and the required TPM operation on a desktop (912.6msec on a 2.2GHz AMD Athlon 64-bit Dual Core processor), it is safe to assume that the situation will be worse on a mobile platform with users frustratingly

noticing their mobile device freeze periodically to allow secure sensing activities to take place.

—Existing mobile devices with sensing capabilities (e.g. smartphones) are as exposed and vulnerable as today's desktops. Our always-on Internet-connected smartphones are already at risk of being usurped by active botnets [Higgins 2010]. A closed box sensing platform, such as one without a connection to the Internet, will make remote attacks on sensory data integrity more challenging.

—Decoupling trustworthy sensing from trusted computing like we have done with the TSP will make trustworthy sensing less invasive and encourage its adoption. Features provided by trusted computing, like TPM-based platform attestation, have been the topic of much debate. Some like Ross Anderson and Bruce Schneier have been vocal critics and consider it to be an invasive technology that encourages software monopolies, facilitates DRM, and compromises privacy [Schneier 2002; Anderson 2003]. For example, since it is necessary to verify the integrity of all software components on a platform to provide integrity assurances about a single one [Sailer et al. 2004], the verifier would effectively know all about the platform's OS and application configuration. Thus, the TPM's platform attestation feature may inadvertently pose a threat to privacy. Isolating the TPM into the closed-box TSP helps alleviate this concern.

—A separate trustworthy sensing device, like our TSP, could expand sensing capabilities. Mobile devices, like smartphones are equipped with only those sensors that help their functionality (e.g. GPS, microphone, accelerometer). However, some crowdsourced sensing applications may require more specialized sensors, such as smog, dust, or chemical pollutant sensors to measure air quality.

6.3. TSP Architecture

The TSP architecture consists of a TPM-capable hardware platform with attached sensors (see Figure 1). Also, a Bluetooth module allows the TSP to communicate with an array of mobile devices supporting the technology.

We achieve closed-box semantics on the TSP by building it using a special-purpose Modified Harvard-architecture sensing platform [Francillon and Castelluccia 2008], and permanently disabling features that may compromise its software integrity. It is widely known, that run-time attacks exploiting memory-related vulnerabilities with the intent of modifying program instructions, have little or no chance of success on such platforms [Francillon and Castelluccia 2008]. Such platforms provide strong physical isolation between executable instructions in program memory, and information in data memory. The program memory on such devices is read-only, and the program counter is not allowed to refer to addresses in data memory. Consequently, program instructions can neither be changed, nor be executed from data memory at run-time. All usual methods to reprogram the TSP—physical or remote—are disabled permanently. Normal communication methods, like the radio, are also disabled. The only connection between the TSP and the outside world is via a Bluetooth channel that provides secure encrypted communication facilities [Bluetooth Special Interest Group 2009; Padgette et al. 2012]. Additionally, the size and format of each STAP packet received via the Bluetooth channel is verfied before being processed any further. These protections shield the TSP from a recent permanent code injection attack against Modified Harvard-architecture platforms [Francillon and Castelluccia 2008].

Given these protections, a physical or remote adversary will not be able to change the preprogrammed firmware on the TSP. Consequently, the on-board TPM need only attest to a firmware version identifier on the device when challenged by a remote entity. Any data sensed by the TSP is signed by the TPM to guarantee its authenticity and integrity. Trust in the data is induced via trust in the platform integrity.

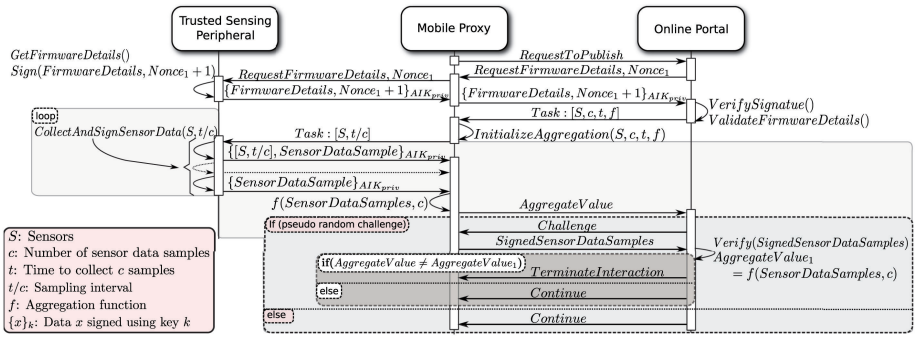


Fig. 2. Secure Tasking and Data Aggregation Protocol (STAP).

6.4. Secure Tasking and Aggregation

Secure tasking involves verifying that the given platform has a TPM (see Section 5), then challenging it to produce a TPM attestation of the platform's firmware, then verifying the attestation is authentic and correctly represents the TSP firmware (e.g. comparing the respective message digests), and finally scheduling the TSP to collect, sign, and send data from a subset of its sensors at regular intervals.

To save energy required for data transmission, and help reduce processing overhead at the portal, the data producer's mobile device (the mobile proxy) may choose to aggregate the raw data from the TSP before forwarding it to the portal. For example, instead of sending raw GPS coordinates from the TSP every 30 seconds, the mobile proxy may send a coarser region covered by those coordinates every five minutes. The problem is that the portal has no way of knowing if the mobile proxy is using the right inputs to the aggregation function (raw data from the TSP) or if it is even computing the correct aggregation function. What is needed is a way for the portal to check the integrity of the aggregation.

We have developed STAP (Figure 2), a protocol that allows the portal to detect whether the untrusted mobile proxy is fabricating the aggregates, as opposed to computing them using the TSP's signed raw data. In the past, algorithms for checking aggregation integrity of data originating from sensors have enabled a large remote sensor network to report aggregation results that were close to the true result [Przydatek et al. 2003]. To achieve complete accuracy, sensor network aggregators needed to send all the raw data input to the aggregation function. Since the transmission costs were assumed to be prohibitive, an approximation was considered acceptable. However, in our scenario, the transmission costs between the mobile proxy (the aggregator) and the portal are not similarly high, so STAP is able to guarantee the accuracy of the aggregation. In addition, STAP allows the portal to tune the number of aggregation results it wants to check. This flexibility enables the portal to trade off resources required to perform integrity checking with the amount of integrity desired. The following paragraphs describe the STAP protocol.

To detect a lying proxy, the portal must know the aggregation function in advance. This can be done using predefined campaigns that specify the data to collect, when to collect that data, and how to aggregate it [Burke et al. 2006; Kapadia et al. 2008]. Then, during the course of receiving data, the portal pseudo-randomly requests the mobile proxy for the latest window of signed raw data used to compute the just-received aggregate (the proxy needs to buffer this data). When the portal receives the raw data from the mobile proxy, it verifies that the TSP signed them, then recomputes the aggregate using the aggregation function, and finally compares the computed aggregate

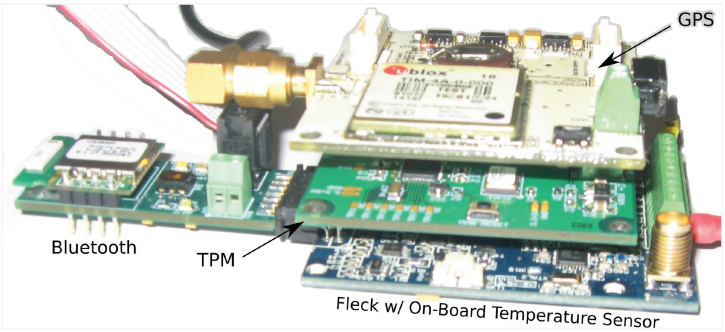


Fig. 3. Trusted Sensing Peripheral with a Bluetooth and GPS module.

with the one already sent. If the comparison fails, the portal can choose to reject further transmissions from this lying proxy. This is similar to the concept of bit-commitment [Chaum et al. 1987], where one commits to a value before it is checked for correctness.

The pseudo-random challenge forces the mobile proxy to guess when it is safe to fabricate an aggregate. Eventually, a lying proxy will guess wrong and get caught. Lying more only causes the proxy to be detected faster, and lying less only delays that outcome. The following equation represents the expected number of aggregates $E(n)$ a portal accepts, when challenging aggregates with probability q, before detecting a proxy lying with probability p.

$$E(n) = \sum_{n=1}^{\infty} n \times (1 - pq)^{n-1} \times pq. \quad (1)$$

We evaluate STAP in Section 8 and show that the experimental results closely match the results obtained using this equation.

7. IMPLEMENTATION

In this section, we discuss our TSP prototype and the implementation of STAP.

7.1. Trusted Sensing Peripheral (TSP)

We use secFleck [Hu et al. 2009] as the foundation of the TSP. secFleck is the portion of the TSP consisting of the Atmel TPM chip (based on v1.2 of the Trusted Computing Group specification [Trusted Computing Group a]), and the Fleck sensor board (see Figure 3).

The Fleck is a sensing platform with 8KB of memory and an 8MHz Atmega micro controller [Sikka et al. 2007]. It houses the TPM module, and is extensible with various sensors. The TSP firmware, including the FleckOS, sensor device drivers, and our application, runs on the Fleck hardware.

Figure 3 shows the TSP. Attached to it, is a Parani-ESD Bluetooth module. The mobile proxy application communicates with the TSP using a local, intermediary Python relay service called `foslisten`. `foslisten` communicates locally using TCP sockets, while with the TSP, it uses a serial-over-Bluetooth channel (Figure 4).

7.2. Online Portal

The online portal is responsible for tasking the TSP, verifying the TSP's platform integrity and the integrity of any data received from it, requesting the mobile proxy to aggregate data if necessary, and pseudo-randomly challenging the mobile proxy to prove its trustworthiness when aggregated data is received. The portal runs as a

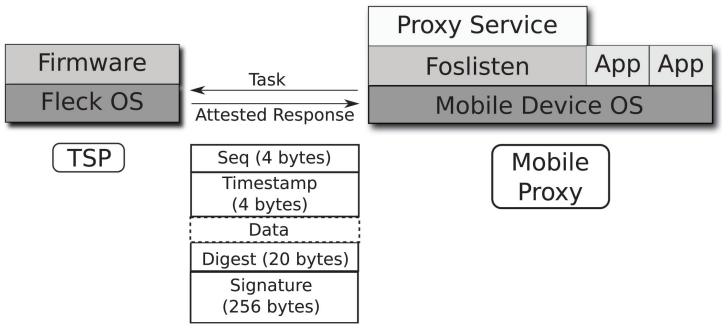


Fig. 4. Data producers on the mobile proxy communicate with the TSP via a Python relay service called 'foslisten'.

standard multi-threaded TCP server on a 2.2GHz Intel(R) Core(TM)2 Duo platform with 1.9GB of memory running Linux kernel version 2.6.32.10.

7.3. Mobile Proxy

The service running on the mobile proxy is responsible for receiving a task from the portal, in turn tasking the TSP via the Bluetooth channel using our custom RPC protocol, retrieving data from the TSP, and forwarding that data to the portal. The mobile proxy is a Nokia N800 tablet with 128 MB of memory running Linux kernel version 2.6.21 on an ARMv6 processor. Although we use the N800, a widely popular tablet, our work is applicable to any smartphone or tablet device.

7.4. Secure Tasking and Aggregation

The details of STAP are shown in Figure 2. Here, we limit our discussion to how the TSP is tasked. The portal first sends a task description $[S, c, t, f]$ to the mobile proxy, where S is the set of sensors to collect data from, c is the number of data samples to collect, t is the time (in seconds) within which to collect the c samples (sampling interval is thus t/c seconds), and f is the aggregation function. S is expressed using a 16-bit mask, allowing sixteen sensors to be tasked simultaneously with the same sampling interval. Each set of data samples is reported in an attested response message signed using AIK_{priv} (see Figure 4), and can be verified by the portal using AIK_{pub}. A mobile proxy modifying the task description will be detected when the portal verifies the one echoed back by the TSP in the first attested response.

8. EVALUATION

We first evaluate the performance of the TSP, and then evaluate STAP in Section 8.2.

8.1. TSP Performance

We analyze the performance of the TSP in terms of time and energy required to perform and transmit data attestations. Then, we discuss other costs involved in building the TSP: code size, memory usage, and monetary cost. The TSP runs on three 1.5V batteries with a 2500mAh capacity each, and is configured with two sensors: an on-board temperature sensor, and an attached GPS sensor (Figure 3).

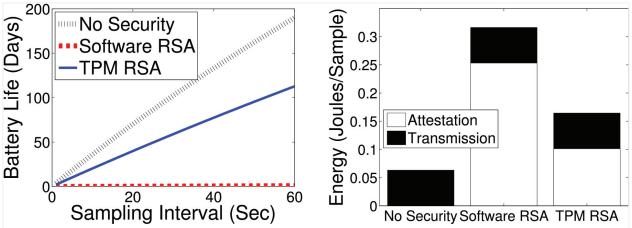
8.1.1. Timing Measurements. The TSP is made to repeatedly perform and return forty attestations, each over a set of temperature data samples varying in size from 2 to 92 bytes at 10 byte intervals. Average time spent performing attestations, along with the 95% confidence interval, is shown in Table I. The small confidence interval is due to the dominant RSA signature operation that is always performed over a fixed size

Table I. Average Time Required to Perform and Transmit an Attestation

Task	Compute Time (sec)	Transmit Time (sec)
Single Attestation	1.72 ($\pm$0.01)	0.3 ($\pm$0.1)

Table II. Average Current Drawn in Various TSP Energy States

Energy State	Current Draw
Idle	80μA
Attesting	50mA
Transmitting	42mA



(a) TSP battery life. (b) TSP energy usage.

Fig. 5. TSP energy measurements.

SHA-1 digest of the data. Table I also shows the average transmission time of the signed data samples. As we can see, the TSP can only sample, attest, and report data at intervals greater than $\approx$2 seconds. This lower limit on sampling and reporting is more than acceptable for existing crowd-sourced mobile embedded sensing systems, most of which, require data samples less frequently than that [Hull et al. 2006; Reddy et al. 2007; Eisenman et al. 2007; Agapie et al. 2008].

8.1.2. Energy Measurements. We measured the TSP's current draw in different energy states using an oscilloscope with an internal resistance of $10M\Omega$ across a 1Ω resistor placed in series with the TSP. Table II shows the average current drawn by the TSP while it was performing attestations (includes sampling), performing transmissions, and while it was idle (no attestations or transmissions).

Using the timing and current draw measurements, we computed the energy consumed by the TSP while attesting and transmitting a 2 byte sample of data at various intervals. Figure 5(b) compares energy consumption of the TSP with the TPM, without any security, and when the TPM's operations are performed in software [Hu et al. 2009]. Hardware attestations end up being half as expensive as those performed in software, because, although they draw more current, they can be computed much faster. However, the TPM requires three times more energy than if there were no security (no attestation, signatures, etc.) at all.

We also computed an estimate of the TSP's battery life. Figure 5(a) shows that when tasked with Attesting and transmitting a 2 byte sample every 30 seconds and remaining idle in-between, the TSP can achieve a battery life of over 80 days. This is quite sufficient for crowd-sourced mobile embedded sensing systems, where participants will carry and eventually recharge these platforms. Also, notice that battery life is nearly double that when all security is in software.

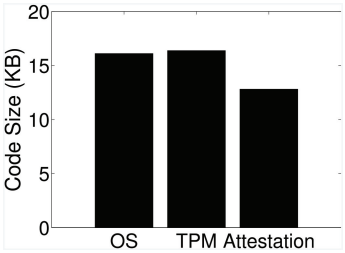


Fig. 6. Compiled code size comparison of various software components on the TSP.

8.1.3. Other Costs. Figure 6 compares compiled code sizes of various software components running on the TSP. The code size of our application is smaller than both the TPM library and the FleckOS. Memory (RAM) used by the firmware is approximately 4.2 KBytes, which is a little over half the available memory (8 KBytes) on the current version of the Fleck. Monetary costs involved are currently high: the cost of the TSP hardware is approximately 300 USD (March, 2010). The TPM chip itself is inexpensive, about 6 USD when purchased in large quantities ($\geq$1000). These monetary costs are high due to the limited scale at which our TSP is currently produced; with economies of scale we believe that the cost could be brought down to lower than 50 USD.

8.2. Secure Tasking and Aggregation Protocol

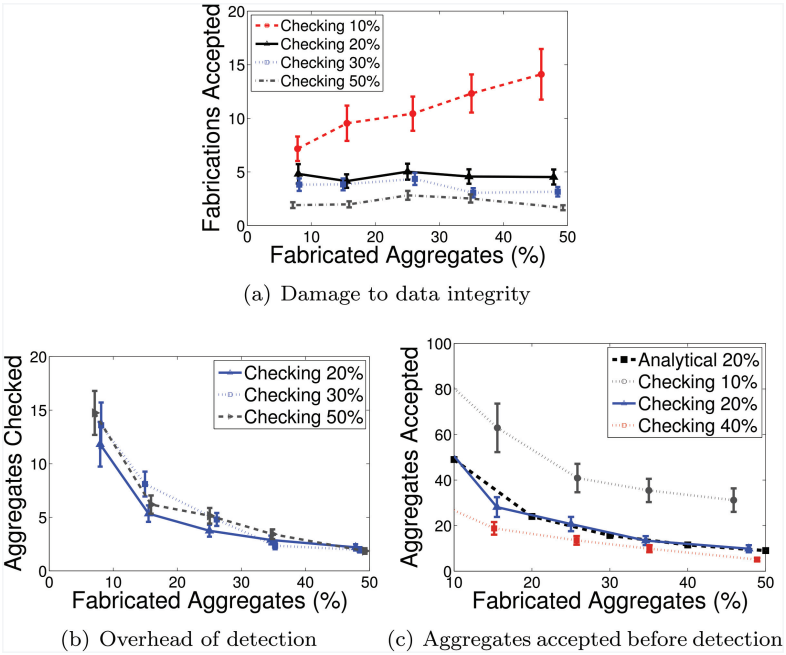
We conducted experiments to answer the following questions about our protocol. (1) How many fabricated aggregate values are accepted by the portal before a lying mobile proxy is detected? (2) How long does it take for the portal to find a lying proxy? (3) What is the overhead of detection?

In each experiment, the mobile proxy is configured to randomly fabricate (or lie about) an aggregate with probability ranging between 1/10 to 1/2 (or 10% to 50% of aggregates). The online portal then pseudo-randomly challenges the integrity of a received aggregate with the same probabilities (Equation (1)). Aggregations are performed on every 10 two-byte samples of temperature data. The aggregation function f performed by the proxy is a mean of those samples. An experiment continues until the portal detects the first fabricated aggregate value. When necessary for clarity, we omit data points that don't add significantly to the information conveyed by the graphs.

Figure 7(a) shows that the portal can quickly and efficiently detect a lying proxy. While pseudo-randomly challenging only 20% of the aggregates, the malicious proxy is detected within the first six fabrications received—no matter how much it lies. The advantage of challenging more than 20% of the aggregates is not significant, thus, this threshold reasonably trades off the integrity of aggregates with the overhead of challenging.

Figure 7(c) shows how long it takes the portal to detect the malicious proxy in terms of the number of aggregates (false or otherwise) accepted before detection. Not surprisingly, the less a proxy lies the longer it takes to detect it. We also plotted Equation (1) with the portal's checking probability q set to 1/5 (or 20% of the aggregates). It can be seen that the analytical plot closely matches the experimental one (Figure 7(c)). Notice also, the large drop in aggregates accepted when challenging 20%, rather than 10% of them.

Figure 7(b) shows the detection overhead in terms of the number of challenges issued before detecting a lying proxy. Surprisingly, the overhead largely depends only on how often the proxy lies. Figure 7(c) provides an intuitive explanation: although the number of aggregates accepted before detection varies significantly, the number of challenges issued does not. This result works in favor of the portal, which can now minimize



(a) Damage to data integrity

(b) Overhead of detection

(c) Aggregates accepted before detection

Fig. 7. Performance of the Secure Aggregation Protocol, STAP.

the number of challenges based on factors like reducing the number of fabrications accepted.

Currently, the portal issues challenges to the proxy uniformly at random. This is because the protocol is currently designed to be general, work for any application or data type, and all data is assumed to be equally important. We believe that more efficient variants of the STAP protocol can be designed by leveraging domain-specific knowledge when deciding how frequently to challenge in large systems. These decisions can be cast in terms of a Stackleberg security framework [Korzhyk et al. 2011].

While outside the scope of this article, we point the readers to recent relevant work that we believe might be fruitfully applied to our problem. Algorithmically, we believe that our problem is similar to recent work on solving the problem of optimizing coast guard patrols, given finite patrolling resources and adversaries that can observe security measures before deciding to attack [An et al. 2013]. Their solution involved the use of Stackelberg Security Games to model the interaction between attackers and a security provider. In the Stackelberg security game framework, the security provider (defender) is modeled as the leader and the attacker is modeled as the follower. The authors, found that the optimal defensive strategies to these games are mixed strategies over different patrolling actions, making decisions unpredictable to the attacker while accounting for the varying importance of different targets and the strategic behavior of the attackers. They model adversaries using Quantal Response (QR) Theory, wherein individuals respond stochastically in games, selecting nonoptimal strategies when the cost of such an error decreases.

9. THREAT ANALYSIS

We now revisit our threat model (Section 4) and describe how our system addresses those threats to the TSP and mobile proxy.

9.1. Threats to the TSP

A majority of the threats are addressed as a consequence of the TPM's special capabilities, namely, a closed-box platform containing a private key and the property of being infeasible to replicate.

—*Software attacks.* Such attacks are mitigated by building the TSP using a modified Harvard-architecture-based platform that provides strict isolation between program and data memory, and permanently disabling all methods that can possibly alter the TSP's firmware.
—*Sybil attacks.* The TPM's sealed private key ties the identity of the TSP (represented by that key) with the hardware platform. Since the TPM cannot be cloned, and its private key cannot be extracted, the adversary has no way to masquerade as a TSP.

9.2. Threats to the Mobile Proxy

The portal combats the malicious data aggregation threat by using STAP to detect a mobile proxy fabricating aggregates. The key idea is that a mobile proxy commits to the aggregate value by the very act of sending it to the portal. The portal then pseudo-randomly verifies the integrity of that commitment. A lying mobile proxy may be able to publish some fabricated aggregates (see Figure 7(a)), but will eventually get caught.

9.3. Threats to Communication

The portal can detect modification or injection of data because an adversary without the TSP cannot fabricate its signatures. Replay attacks can be detected because each data sample generated by the TSP has an incremental sequence number.

10. LIMITATIONS AND FUTURE WORK

We recognize that the use of the TPM cannot protect against all failure modes. For example, sensor measurements may be inherently corrupt, sensors may be damaged, or a sensor's environment may be doctored. It is also likely that not every user has a Trusted Sensing Peripheral. In this case, the crowd-sourced mobile embedded sensing systems could still collect data from untrusted sensing platforms, but then use the data from trusted peripherals to calibrate or validate the untrusted data. We plan to address a solution along these lines in future work.

The types of aggregation functions addressed in this article work on discrete windows of data. Thus, calculating metrics like the median would require a different aggregation protocol. Nonetheless, such a protocol could easily be integrated within our current framework.

The TSP may duplicate some functionality of built-in smartphone sensors such as audio and GPS, however, it also enables the addition of a wide range of new sensors such as CO, CO_2, humidity, temperature, seismic, and medical sensors. For example, carbon monoxide studies in Ghana used an additional device attached to the phone [Paulos et al.]. Furthermore, the TSP can be connected to a legacy mobile device, so that it is possible to deploy a trustworthy crowd-sourced sensing application without the cooperation of smartphone OEMs such as Apple or Nokia.

11. CONCLUSION

This article presented the design and implementation of a Trusted Sensing Peripheral (TSP) to provide data authenticity and integrity for crowd-sourced mobile embedded sensing systems. The TSP has built-in sensors, and a Trusted Platform Module (TPM) that helps it resist software and Sybil attacks to its platform. The TPM attests to the integrity of published data at the source, and this attestation is verified at a remote

server. The TSP is energy-efficient, with a battery life of over 80 days when collecting a temperature sample every 30 seconds.

Our secure data aggregation protocol STAP allows an untrusted intermediate mobile proxy to aggregate the TSP's raw signed sensor readings, while allowing the data portal to detect a malicious proxy that fabricates those aggregates. The portal can detect a lying proxy with little overhead and within the first six fabricated aggregates received.

To the best of our knowledge, this is the first work to present a practical implementation of a trusted platform-based approach using a physical TPM device applied to the data integrity problem in crowd-sourced sensing, and a secure and trusted data aggregation protocol. This article shows that such an approach is practical, computationally feasible, and energy-efficient, especially with the introduction of new system-on-chip solutions such as those from Texas Instruments [TI 2012]. However, the problem with the approach for now is the high cost of the TSP, high which every user needs to have.

REFERENCES

Advanced Micro Devices. SVM: AMD's virtualization technology. www.xen.org/files/xs0106_amd_virtualization.pdf.

E. Agapie, G. Chen, and D. Houston et al. 2008. Seeing our signals: Combining location traces and Web-based models for personal discovery. In *Proceedings of ACM HotMobile.* 6–10.

B. An, F. Ordez, M. Tambe, E. Shieh, R. Yang, C. Baldwin, J. DiRenzo, K. Moretti, B. Maule, and G. Meyer. 2013. A deployed quantal response-based patrol planning system for the U.S. Coast Guard. *Interfaces* 43, 5, 400–420.

R. Anderson. 2003. 'Trusted Computing' frequently asked questions. http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html.

Atmel Corporation. The Atmel trusted platform module. www.atmel.com/dyn/resources/prod_documents/doc5128.pdf.

N. Baughman and B. Levine. 2001. Cheat-proof playout for centralized and distributed online games. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*.

Bluetooth Special Interest Group. 2009. Core version 3.0 + HS. https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=174214.

J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. Srivastava. 2006. Participatory sensing. In *Proceedings of the ACM Sensys Workshop on World-Sensor-Web*.

D. Chaum, I. Damgård, and J. van de Graaf. 1987. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *Advances in Cryptology*, Springer, 87–119.

CNN. CNN iReport - Share your story, discuss the issues with CNN.com. http://www.ireport.com/.

P. Denantes, F. Bénézit, P. Thiran, and M. Vetterli. 2008. Which distributed averaging algorithm should I choose for my sensor network? In *Proceedings of the 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM).* 986–994.

R. Dingledine, N. Mathewson, and P. Syverson. 2004. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium*.

J. Douceur. 2002. The Sybil attack. In *Proceedings of the IPTPS Workshop.*

A. Dua, N. Bulusu, W. Feng, and W. Hu. 2009. Towards trustworthy participatory sensing. In *Proceedings of the 4th USENIX Workshop on Hot Topics in Security (HotSec).*

S. Eisenman, E. Miluzzo, N. Lane, R. Peterson, G. Ahn, and A. Campbell. 2007. TheBikeNet mobile sensing system for cyclist experience mapping. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*. ACM, 87–101.

A. Francillon and C. Castelluccia. 2008. Code injection attacks on Harvard-architecture devices. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*. 15–26.

S. Ganeriwal, L. Balzano, and M. Srivastava. 2008. Reputation-based framework for high integrity sensor networks. *ACM Trans. Sens. Netw.* 4, 3.

T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. 2003. Terra: A virtual machine-based platform for trusted computing. *ACM SIGOPS Oper. Syst. Rev.* 37, 5, 206.

P. Gilbert, J. Jung, K. Lee, H. Qin, D. Sharkey, A. Sheth, and L. P. Cox. 2011. Youprove: authenticity and fidelity in mobile sensing. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys)*. 176–189.

K. Higgins. 2010. Smartphone Weather App Builds a Mobile Botnet. http://www.darkreading.com/insiderthreat/security/client/showArticle.jhtml?articleID=223200001.

W. Hu, P. Corke, W. C. Shih, and L. Overs. 2009. secFleck: A public key technology platform for wireless sensor networks. In *Proceedings of EWSN*. 296–311.

B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. 2006. Cartel: A distributed mobile sensor computing system. In *Proceedings of ACM SenSys*. 125–138.

Intel Corporation. Intel trusted execution technology. http://www.intel.com/technology/security/.

A. Kapadia, N. Triandopoulos, C. Cornelius, D. Peebles, and D. Kotz. 2008. Anony- Sense: Opportunistic and privacy-preserving context collection. In *Lecture Notes in Computer Science*, vol. 5013, 280.

D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. 2011. Stackelberg vs Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *J. Artif. Int. Res.* 41, 2, 297–327.

N. Lathia, K. K. Rachuri, C. Mascolo, and P. J. Rentfrow. 2013. Contextual dissonance: Design bias in sensor-based experience sampling methods. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing* (*UbiComp*). 183–192.

J. McCune, B. Parno, A. Perrig, M. Reiter, and H. Isozaki. 2008. Flicker: An execution infrastructure for TCB minimization. In *Proceedings of ACM SIGOPS/EuroSys*. 315–328.

S. Nath, J. Liu, J. Miller, F. Zhao, and A. Santanche. 2006. SensorMap: A Web site for sensors world-wide. In *Proceedings of ACM SenSys*. 373–374.

openssl.org. Openssl: The open source toolkit for ssl/tls. http://www.openssl.org/.

J. Padgette, K. Scarfone, and L. Chen. 2012. Guide to Bluetooth Security. http://csrc.nist.gov/publications/nistpubs/800-121-rev1/sp800-121_rev1.pdf.

E. Paulos, I. Smith, and R. Honicky. Participatory urbanism. http://www.urban-atmospheres.net/ParticipatoryUrbanism/index.html.

R. A. Popa, H. Balakrishnan, and A. J. Blumberg. 2009. Vpriv: Protecting privacy in location-based vehicular services. In *Proceedings of the USENIX Security Symposium*. 335–350.

B. Przydatek, D. Song, and A. Perrig. 2003. SIA: Secure Information Aggregation in Sensor Networks. In *Proceedings of ACM SenSys*. 255–265.

V. Rastogi and S. Nath. 2010. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 735–746.

S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen. 2007. Image browsing, processing, and clustering for participatory sensing: Lessons from a DietSense prototype. In *Proceedings of ACM SenSys*. 13–17.

R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn. 2004. Design and implementation of a TCG-based integrity measurement architecture. In *Proceedings of the USENIX Security Symposium*. 223–238.

B. Schneier. 2002. Palladium and the TCPA. http://www.schneier.com/crypto-gram-0208.html#1.

A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. 2005. Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems. In *Proceedings of ACM SIGOPS* 39, 5, 1–16.

A. Seshadri, A. Perrig, L. Van Doorn, and P. Khosla. 2004. SWATT: Software-based attestation for embedded devices. In *Proceedings of the IEEE Symposium on Security and Privacy*. Citeseer, 272–282.

A. Sharma, L. Golubchik, and R. Govindan. 2007. On the prevalence of sensor faults in real-world deployments. In *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. 213–222.

E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. 2011. Privacy-preserving aggregation of time-series data. In *Proceedings of NDSS*. Vol. 2. 4.

P. Sikka, P. Corke, L. Overs, P. Valencia, and T. Wark. 2007. Fleck: A platform for real-world outdoor sensor networks. In *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*. 709–714.

F. Stajano and R. Anderson. 2000. The resurrecting duckling: Security issues for ad-hoc wireless networks. *Lecture Notes in Computer Science,* vol. 1796, 172–182.

The H. Security. 2010. Hacker extracts crypto key from TPM chip. http://www.h-online.com/security/news/item/Hacker-extracts-crypto-key-from-TPM-chip-927077.html.

TI. 2012. Wireless Connectivity - ZigBee (IEEE 802.15.4/ZigBee PRO) - CC2538 - TI.com. http://www.ti.com/product/cc2538.

Trusted Computing Group a. About TCG. http://www.trustedcomputinggroup.org/about_tcg.

Trusted Computing Group b. Platform reset attack mitigation specification, Version 1.0. http://www.trustedcomputinggroup.org/resources/pc_client_work_group_platform_reset_attack_mitigation_specification_version_10/.

Trusted Computing Group c. Trusted platform module (TPM) specifications. http://www.trustedcomputinggroup.org/developers/trusted_platform_module/specifications.

Waze. Free GPS navigation with turn by turn directions. http://www.waze.com/homepage/.

S. Zhu, S. Setia, S. Jajodia, and P. Ning. 2004. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *Proceedings of the IEEE Symposium on Security and Privacy*. 259–271.

Received April 2013; revised November 2013; accepted February 2014