I Am a Sensor, and I Approve This Message

Stefan Saroiu and Alec Wolman
Microsoft Research

Abstract: Despite the popularity of adding sensors to mobile devices, the readings provided by these sensors cannot be trusted. Users can fabricate sensor readings with relatively little effort. This lack of trust discourages the emergence of applications where users have an incentive to lie about their sensor readings, such as falsifying a location or altering a photo taken by the camera.

This paper presents a broad range of applications that would benefit from the deployment of trusted sensors, from participatory sensing to monitoring energy consumption. We present two design alternatives for making sensor readings trustworthy. Although both designs rely on the presence of a trusted platform module (TPM), they trade-off security guarantees for hardware requirements. While our first design is less secure, it requires no additional hardware beyond a TPM, unlike our second design. Finally, we present the privacy issues arising from the deployment of trusted sensors and we discuss protocols that can overcome them.

1. INTRODUCTION

One important factor contributing to the rapid growth of smartphones is the incorporation of more sensors into these devices. To take one example, the latest iPhone has a GPS chip, an accelerometer, a digital compass, a proximity sensor, an ambient light sensor, a microphone, and a camera. This variety of sensors has given rise to a huge number of innovative mobile applications. In the future, manufacturers may incorporate even more sensors into smartphones, such as fingerprint readers, radiation detectors, water quality sensors, and personal health sensors.

Today, it is relatively easy for malicious applications to fabricate or lie about readings from these sensors. For example, users can easily lie about their locations by fabricating readings from GPS sensors, or they can modify the pictures taken by their cameras. As a result, developers are reluctant to build mobile applications where users have an incentive to cheat. To address this problem, we propose using hardware support for trusted computing to make the data obtained from sensors trustworthy, thereby vastly reducing the possibility of users cheating.

One popular class of mobile applications is *participatory sensing*: applications that create databases of information based on inputs collected from individual mobile users. Examples include public positioning systems [22], maps of Wi-Fi availability [26], and maps of Swine Flu outbreaks [2]. One challenge that all such applications face is that of *data pollution*: malicious users can manipulate their contributions to "pollute" the database, either by spoofing their location or other sensed data. Research projects have already started to characterize the extent of damage these security attacks can have on a database [23]; an even more alarming possibility is that of a worm that can infect mobile devices causing them to upload polluted sensor readings.

In this paper, we present a variety of applications that would benefit from hardware and software support for trustworthy sensing. Although our primary focus is on mobile applications, we do not limit ourselves solely to mobile applications. We examine different design and implementation alternatives for providing trusted sensors by leveraging trusted hardware support. Although today's trusted computing hardware, such as a trusted platform module (TPM) [25] and Intel's trusted execution technology (TXT) [15], is only available for desktop and server machines, we expect that such features will soon be integrated into mobile devices. In this work we describe two design alternatives: one that uses a system-wide TPM to increase the trustworthiness of sensor readings, and another that directly integrates TPM functions into the sensors themselves. Our goal is to put forward different alternatives to begin the discussion on how to provide trusted sensors.

In previous work we described location proofs, a trusted infrastructure for determining device location, as well as a set of applications that would benefit from this infrastructure [21]. One drawback of location proofs is that it requires significant infrastructure deployment; it cannot be deployed without modifying current Wi-Fi access points. While figuring out how to remove these barriers to deployment, we realized that a much simpler deployment approach would be to leverage previous work on trusted computing hardware for desktops and servers, by making use of TPM-like hardware support to secure readings from a GPS radio. This paper grew out of that realization: in fact, adding such functionality to smartphones would enable a wide variety of trusted sensors.

To illustrate how trusted sensors work, let's consider the example of a participatory-sensing system that relies on users transmitting photos along with their location information. With trusted GPS and camera sensors, a photo is combined with a GPS reading and a timestamp, and then signed with a private key specific to the mobile device. The goal of this architecture is to ensure that untrusted software running on the mobile device cannot interfere with reading the sensor values. Once the data is uploaded to a geo-tagging website, the signature is verified and then the photo is added to the collection along with its time and location. This makes it much harder for a rogue user to upload incorrect information into the database.

Privacy is a significant concern for any service where uploaded data can be tied back to a particular mobile device. We know of two possible approaches to alleviate this concern. First, we believe that the trusted sensor software must be architected to allow users to easily remove signatures from the sensor readings. Because unsigned readings can easily be fabricated, they reduce the privacy risks. The user can then *choose* whether or not to provide the signature along with the sensor reading to either a local application or a remote cloud service. The second approach to alleviate privacy concerns is to use cryptographic protocols that protect users' privacy. One possibility is to use the TPM's anonymous attestations to provide users with anonymity. Another approach is to use zero-knowledge protocols to prove that a sensor reading is signed by a

HotMobile 2010, February 22-23, 2010, Annapolis, MD, USA.

TPM. Rather than pass along a signed sensor reading from a mobile device to a remote service, zero-knowledge proofs allow a mobile client to prove to a remote server that they possess a signed sensor reading without directly revealing the signature.

In this paper, Section 2 provides a brief primer on existing hardware support for trusted computing. Section 3 then describes a set mobile applications made possible by trusted sensors. Section 4 outlines our two designs for supporting trusted sensing. Section 5 describes several privacy issues arising from the use of trusted sensors. Finally, we briefly summarize related work (Section 6) and conclude (Section 7).

2. BRIEF TPM PRIMER

A Trusted Platform Module (TPM) is a specialized piece of hardware, shipped with many of today's desktops and laptops, that offers three trusted computing primitives:

• **Remote Attestation:** enables users to remotely attest that a machine booted a certain hardware and software configuration.

• **Sealed Storage:** protects data by binding it to a particular TPM and software configuration in a way that can only be accessed by the same combination of hardware and software.

• **Secure Boot:** ensures that the machine can only boot a certain hardware and software configuration.

A TPM offers a number of platform configuration registers[1] (PCRs) and two related instructions: reading a PCR and "extending" a PCR. Extending a PCR is a way of updating the register; an extend call takes as input a piece of data, hashes it, and performs an "OR" operation with the register's current value. Thus, the value found in a PCR register is a function of all the previous hash values (i.e., a running hash) with which the PCR has been "extended". All PCR registers are cleared upon a reboot.

When a TPM-enabled machine boots, it calculates a hash of the BIOS code and extends one of the PCR registers with the result, a process referred to as a "measurement". The BIOS code then executes and before passing control to the loader, the BIOS hashes the loader code and extends the same PCR register (i.e., measures the loader), computing a running hash. Each step during the bootup is performed the same way: the next piece of code ready to run is first hashed and the hash value is used to extend a PCR register. This running hash effectively creates a secure chain of trust, where each measurement authenticates a step during bootup.

Remote Attestation: A remote attestation is a piece of data containing the values of certain PCRs signed by the TPM. The TPM's signature ensures the PCRs' integrity; a verifier can then check whether the PCR values match those that should be obtained when booting the "correct" software configuration.

Sealed Storage: With sealed storage, data is stored encrypted with a key that is a function of the values stored in the PCRs. The encryption key is protected by the TPM and released only if the PCRs hold the same values as when the data was sealed. This ensures data is unsealed by the same software that sealed it in the first place.

Secure Boot: For secure boot, the OS image is encrypted with a key that has been sealed by the TPM. When booting up, the TPM measures each boot step from the BIOS to the loader. Once the OS is ready to load, the TPM checks whether the values found in the PCR registers match the ones that the OS image's encryption key has been sealed with. This ensures that none of the boot code has been tampered with; in this case, the TPM then releases the key needed to decrypt the OS image. Once decrypted, the OS image is loaded to finish the secure boot process.

[1]The number of registers is different for each TPM version; newer versions have 24 registers.

2.1 TXT Extensions

Recently, Intel and AMD have both enhanced their trusted computing primitives with additional extensions, referred to as Intel's Trusted eXecution Technology (TXT) [15] and AMD's Secure Virtual Machine (SVM) [1]. With these extensions, the TPM contains a new class of PCRs, called dynamic PCRs, and a new CPU instruction, called *SKINIT*[2]. The goal of these extensions is to perform code measurements without needing to establish an entire chain of trust starting from the BIOS (i.e., a running hash).

SKINIT takes as input a physical memory address where a small loader resides. When invoked, the dynamic PCR registers are cleared, DMA access to the physical pages storing the code for the small loader is disabled, interrupts are turned off, and debugging is disabled. The TPM performs a measurement of the loader before the processor starts to execute it. As the loader executes, it loads an application and measures it with the TPM; once the measurement is complete and the result is stored in the dynamic PCRs, the loader starts executing the application.

2.2 Anonymous Attestation

The goal of anonymous attestation is to protect a TPM's identity while still enabling a verifier to check that a remote attestation has been signed by a valid TPM (i.e., without revealing the TPM's identity). A version 1.1 TPM can provide anonymous attestations by using a separate Attestation Identity Key (AIK) for each verifier and relying on a privacy certification authority (privacy CA). A TPM generates an AIK key, signs it, and sends it to the privacy CA. The privacy CA checks the signature and returns a certificate. The TPM uses this certificate to anonymously sign attestations. A verifier can contact the privacy CA to check the validity of the TPM certificate.

Although anonymous attestations protect the identity of a TPM, they suffer from two drawbacks [3]: (1) the third-party privacy CA must be involved during an attestation issue and verification, and (2) the TPM's identity is revealed to the third-party during each attestation putting tremendous power in the hands of the privacy CA. The current TPM specification (version 1.2) implements an additional anonymous attestation scheme, called Direct Anonymous Attestation (DAA), that does not need a privacy CA [3].

3. APPLICATIONS

In this section, we describe a number of potential applications that can be enabled by the deployment of trusted sensors. The common theme across all these applications is that users have an incentive to lie about the readings of their sensors.

3.1 Location Proofs

In our previous work, we presented six applications that could benefit from location proofs – a trusted infrastructure that enables mobile devices to determine their location [21].

• Store discounts for loyal customers: offering discounts to the customers who visit a store repeatedly.

• Green commuting: rewarding people who leave their cars at home and instead walk, bike, or commute by bus to work.

• Location-restricted content delivery: providing fine-grained location information about users accessing sites that provide content which is location-specific or subject to local copyright laws.

[2]SKINIT is the AMD instruction; on an Intel CPU, the instruction is called GETSEC[SENTER].

• Reducing fraud on online auctions: increasing confidence in a financial transaction by establishing a buyer or seller's location.

• Voter registration: demonstrating the physical presence requirement common to many forms of elections in the US.

• Police investigations: allowing a suspect in an investigation to produce an alibi.

A trusted GPS sensor providing a signed reading of a user's location combined with a timestamp would eliminate the need for location proofs. As a result, the above applications would benefit from a trusted GPS sensor. Because these applications are already described in-depth in [21], we refrain from revisiting them here.

3.2 Participatory Sensing

In participatory sensing applications, users upload individual sensor readings to a central database, typically indexed by location. By spreading the work of data collection across a large pool of users, participatory sensing applications can rapidly build vast repositories of useful information. Examples of participatory sensing applications include the locations of Wi-Fi access points [22], traffic and road conditions [18], monitoring air quality [19], understanding how swine flu is spreading across the country [2], and geo-tagged photos of buildings and landmarks [12]. As new sensors are developed for mobile devices (e.g., a radiation sensor), we expect new participatory sensing applications to emerge based on these sensors [20].

One common problem faced by participatory sensing applications is that of *data pollution*: malicious users can upload forged data to "pollute" the database with false information. The Internet's lack of strong authentication mechanisms exacerbates this problem; a single user can create vast amounts of pollution. Data pollution prevention and detection for participatory sensing applications is an ongoing research problem [16].

Efforts to reduce data pollution could benefit from trusted sensors in two ways. First, an application could require each participant to use their TPM to sign their data samples, which would limit the amount of data pollution from a single user. Second, applications that collect sensor readings (e.g., GPS-based locations or photos) could require their participants to use a trusted sensor when uploading a reading. This would drastically reduce the likelihood that the sensor reading has been tampered with.

However, not all participatory sensing applications collect *only* sensor readings. For example, a collaborative application that maps the spread of the influenza virus requires each user to upload: (1) the user's location based on a sensor reading, and (2) the user's health information (i.e., whether or not the user is infected) based on user input. A trusted GPS can help ensure the accuracy of the location data, but trusted sensors cannot prevent the user from providing incorrect information.

3.3 Online Authentication

Although today's mobile phones typically do not include a fingerprint reader, we envision a number of uses for these sensors. A trusted fingerprint reader could provide a simple way for users to authenticate to online websites. While password-based systems are by far the most common form of online authentication today, mobile phone users might prefer a fingerprint reader as an alternative to passwords, especially given the difficulty of typing on today's smartphone keyboards. Fingerprint readers are relatively convenient and they avoid the risk of users forgetting their passwords. These properties could be quite appealing to many Internet users, and this may motivate websites to offer this form of online authentication.

3.4 User Presence Detection

The ability to distinguish between an activity performed by a person versus one performed by a program has many uses related to Internet security. A recent project [13] has proposed addressing spam, DDoS attacks, and click fraud, using trusted computing to detect that a human is responsible for generating keyboard or mouse events, and linking this to Internet activities such as sending e-mail or surfing the Web. Trusted sensors offer an alternative solution for this class of problems, as we now describe.

3.4.1 Solving CAPTCHAs

Web sites currently use CAPTCHAs for certain operations to distinguish between real users and automated programs. These operations include user registration and resetting one's password. Despite their widespread use, CAPTCHAs are becoming increasingly difficult for users. As CAPTCHA-breaking programs become more sophisticated, Web sites are forced to increase the difficulty of extracting text from CAPTCHAs, which in turn makes it much harder for users to decipher them. Another limitation is that people with certain disabilities cannot solve CAPTCHA challenges.

Trusted sensors provide an alternative to CAPTCHAs by increasing the likelihood that requests are made on behalf of real users. For example, a site could require customers to use a trusted fingerprint reader to create a new account, although a trusted fingerprint reader would uniquely identify the user making the request. There are also other solutions with better privacy properties that use trusted sensors. For example, a Web site could show a random number (i.e., a "nonce") to the user and ask them to read back the number using a trusted microphone. The trusted microphone adds a timestamp to the voice sample before signing it and passing to the Web site. The site can use speech recognition to verify that the spoken number matches the challenge and that the timestamp is accurate (e.g., to eliminate "replay" attacks). Another solution for CAPTCHAs would be to use a trusted proximity sensor.

3.4.2 Fighting Spam, DDoS, and Click Fraud

Trusted sensors can also enable tagging e-mail messages or Web requests with evidence that they were generated by a human. For example, an e-mail program could ask the user to use a trusted fingerprint reader whenever sending an e-mail. Similarly, a trusted proximity sensor could tag each Web request with proof that a human is "near" the mobile device making the request.

3.5 Authentic User-Generated Content

News companies are increasingly asking the general public to contribute photos and videos when present at a "news-worthy" event. Such user-generated content is often included in breaking news stories on TV and on the Web. One problem news sites face is verifying that this user-generated content is authentic and has not been altered or manipulated. For example, videos can be modified to include video frames gathered elsewhere, and photos can be "photoshopped" to add or remove people at an event.

A trusted camera integrated into smartphones can reduce the chance that user-contributed content has been manipulated. A photo or video taken with a trusted camera would embed a signed location and timestamp in the content; news companies can increase their confidence in the legitimacy of submitted content by checking these location and timestamp readings.

3.6 Car Sensors

As cars become more sophisticated, car manufacturers can add new sensors to help prevent drunk driving, to help parents monitor their children's driving, or to prevent teenagers from driving too

fast. In all these scenarios, drivers have an incentive to cheat by altering or fabricating a sensor reading. Trusted sensors would raise the bar for such attacks by providing evidence that the sensed data has not been tampered with.

3.7 Returning a Borrowed Item

One issue that often arises when returning a borrowed item (e.g., a book from the library or a car rental) is determining who is responsible for any damage to the item being returned. A trusted camera can help determine if the problem was present before the item was borrowed. For example, a customer can take pictures of the rental car before signing the rental agreement, and then prove that they are not responsible for any pre-existing scratches or dents.

3.8 Hazardous Noise

People subjected to unhealthy levels of noise can suffer permanent hearing damage. To prevent this, people may need to resort to the courts to obtain noise-level ordinances or to shutdown activity that generates unhealthy levels of noise. A trusted microphone could allow people to provide concrete evidence of harmful noise levels.

3.9 Documentary Evidence of Crime Scenes

Ensuring that evidence is admissible in a court of law depends in part on documenting how that evidence was handled between the crime scene and the court room. There are known cases where crime scene investigators or prosecutors have tampered with the evidence collected at a crime scene. When tampering is detected the evidence can no longer be used, and when it is not detected, innocent people can suffer unfortunate consequences. Trusted sensors could reduce both of these risks by making it difficult to tamper with evidence from crime scenes. For example, a trusted camera would make it harder to alter photos, and a trusted microphone would make it harder to alter an interview with a witness.

3.10 Monitoring Energy Consumption

Utility companies perform periodic sensor readings to determine how much water, electricity, or gas a household consumes. Today, users must simply trust that these readings are accurate. Trusted sensors can empower the consumer to require the utility companies to provide the trusted sensor readings along with their bills. Consumers can use these readings to prevent over-charging (either deliberate or accidental).

3.11 Encounter Proofs

People can use a trusted short-range wireless interface (e.g., Bluetooth) to construct an *encounter proof*: a proof that they encountered a certain other person also carrying a Bluetooth-enabled device. The role of the trusted wireless interface is to measure the radio characteristics of the other discovered device and to sign them to prevent further tampering. Recent work [4] has shown that variations in radio transmitter hardware effectively provide a unique per-device signature. This signed radio signature is effectively an encounter proof: it can be used to prove the encounter was with a specific individual (i.e., the one carrying the radio whose signals are present in the encounter proof) and to prove repeated encounters with the same individual.

4. TWO DESIGNS

This section presents two different designs for building trusted sensors. The first design assumes no additional hardware beyond a TPM chip on the motherboard of a mobile device. To make a sensor's reading trusted, this design relies on an isolated attester: a small piece of trusted code that runs isolated from the device's kernel and applications. The second design incorporates trusted computing primitives into sensors to enable sensors to sign their readings. This ensures that any tampering with a sensor reading can be detected because it will invalidate the signature.

In this paper, we deliberately chose to present two designs because of their different trade-offs. The first design is software-based: it has weaker security guarantees but it also has a lower barrier to deployment because it does not require significant hardware modifications. The second design makes it more difficult to tamper with a sensor's readings, but it has a higher deployment cost because it requires more substantial hardware modifications. We believe that both designs are valuable depending on the timeframe: the first design is an easier short-term solution, whereas the second design is a more secure long-term solution.

4.1 Design # 1

The main challenges of this design are two-fold: (1) reading the sensor using an non-malicious piece of code that does not tamper with the reading; and (2) ensuring that the reading cannot be tampered with while being sent to an application or to the cloud. We accomplish these two goals through a combination of virtualization and trusted computing.

In our virtualization design, the user's software environment runs as a guest virtual machine (VM) (e.g., a domU VM in Xen or a child partition in Microsoft's Hyper-V). The root VM is inaccessible to the user. Its role is to read the mobile device's sensors, use the TPM to sign them, and provide these signed readings to the guest VM. The hypervisor virtualizes the remaining devices (i.e., all devices other than the trusted sensors) so that the guest VM can still access them. When one needs a sensor reading, the user makes a hypervisor call to obtain a signed reading from the root VM. As long as the hypervisor and the root VM remain uncompromised, the user has no way to directly read the sensors. Our design uses the TPM secure boot features to ensure that the mobile device boots the correct software configuration. It also relies on an IOMMU to avoid DMA attacks mounted from the guest VM because such DMA attacks could bypass the hypervisor's isolation.

One key requirement for trusted sensing applications is the ability to combine sensor readings. In particular, knowing that a standalone sensor reading is a valid reading is much less useful than knowing when and where that reading was obtained. We propose using time as the common element to enable combining readings. When the user requests a secure reading, it invokes a hypercall that will then read the sensor and the clock. The sensor and the clock readings are then signed by the TPM and then passed back up to the application. Once signed by the TPM, these readings cannot be modified. This ensures that even though the reading is now handled by untrusted software running in the user's VM, the software cannot tamper with the reading without being detected. Incorporating time into each signed reading enables a remote service to combine multiple sensor readings: a signed photo plus timestamp t1 generated by a specific TPM can be combined with a signed GPS location plus timestamp t2 signed by the same TPM. The verifier can check that the signing TPM is the same, and that the time difference between t1 and t2 is less than a threshold.

The security of our first design can be compromised through hardware-based attacks. At boot time, the TPM is able to check only that the BIOS, the boot loader, the attester, and the OS have not been compromised. If the attacker has compromised the hardware by making the sensor faulty (i.e., provide incorrect readings) or by adding a modified sensor that can provide forged readings, our design does not detect such attacks: the attester will still sign

the faulty readings. If the attacker can alter the device's clock, the attacker could make our design generate sensor readings with incorrect timestamps. One way to protect against such attacks is to leverage additional hardware support, such as in our second design that incorporates trusted computing semantics into the sensors.

4.2 Design # 2

For this design, we envision TPM-like functionality being integrated into each individual sensor device. This enables each sensor reading to be signed by the I/O device that generated the reading, independent of the software configuration of the smartphone. Thus, we can provide signed raw sensor readings without relying on the TPM's trusted boot features. However, three key challenges remain for this design: 1) raw sensor readings may be too low-level for the application's desired semantics, 2) a remote service needs the ability to know which sensors are associated with a particular user's device, and 3) applications need to tie together multiple sensor readings on the same mobile device, just as with our first design. We now consider each of these challenges.

To enable application-specific code that uses multiple raw sensor readings as input and provides a high-level result to a cloud service, we leverage the TXT extensions on the mobile device CPU. The cloud service provides a small piece of trusted code to the mobile device that combines the raw sensor readings, and this code performs signature verification on each raw sensor reading. The TXT extensions enable the cloud service to verify that the mobile device was actually the trusted sensor processing code.

To enable a remote service to know which sensors are associated with a particular device, we use a device "registration" process. The cloud service generates a nonce, and the mobile device's TPM and each of its trusted sensors signs the nonce along with a timestamp using registration keys. The cloud service then stores the registered public keys for the device's TPM and for each of its trusted sensors. Therefore, as long as the mobile device is not compromised at device registration time, the cloud service can now know (and identify) which sensors a user's mobile device contains.

To enable tying together multiple sensor readings, we would like to use time as the common element, as we did with our first design. However, incorporating a secure clock into each hardware sensor seems challenging. Instead, we propose that when each sensor generates a signed reading, it contacts the motherboard TPM and increments a well-known secure counter (one that is initialized at device registration time). The sensor concatenates the TPM-signed counter value with the current sensor reading, and signs both.

Now, lets consider an example of how to use this infrastructure to generate a trusted sensor reading where both the time and the location of the sensor reading are known. At time t1, a secure time and location reading is obtained from the mobile device's GPS unit: the secure counter value included in this reading c1. At time t2, the secure sensor reading is obtained, and the secure counter value in this signed reading is c2 (c2 > c1). At time t3, another secure time reading is obtained from the GPS and it includes counter value c3 (c3 > c2). All three signed readings are now provided to a cloud service for verification. The cloud service verifies: 1) c3 > c2 > c1, 2) all three counters are signed by the same TPM, 3) all three sensors are part of the same mobile device (from the device registration described above) 4) that the time difference between t1 and t3 is less than some modest threshold (e.g., less than a minute). Note that this scheme relies upon GPS signals which are not easily available indoors. However, we can use GPS to bound the start and end period of when a sensor reading was taken, and we can use the secure counter to combine this information with periodic reading from the mobile device's internal clock.

This design requires that sensors perform one additional computational step beyond the previous design: sensors must sign their readings and this adds extra overhead to a sensor's performance. The overhead can be small if the sensor is equipped with a relatively powerful processor; however, this presents a trade-off: a faster processor is more energy hungry and more costly. We plan to investigate this performance versus cost trade-off in a future prototype implementation.

5. PRIVACY ISSUES

Privacy is an important concern to our design of trusted sensors because their readings are signed with a TPM's private key. To alleviate this concern, one possibility is to allow users to obtain unsigned readings from their trusted sensors. While this would eliminate the potential of a privacy breach, it also invalidates the benefits of trusted sensors. In particular, it leaves users unable to demonstrate that they have not tampered with their sensors' readings. In this section, we present two possibilities to reduce the privacy concerns of trusted sensors while retaining some of their benefits.

We believe that users will have two main privacy concerns when using trusted sensors: 1) **Anonymity:** Users want to be able to prove they have not tampered with their sensor readings *without* revealing their identities, and 2) **Non-transferability:** Users want to ensure that no one else can prove the validity of their trusted sensor readings. Thus, when an application verifies a sensor reading, it cannot transfer enough information to others that allows them to verify it. We now present how to incorporate these privacy goals in the design of trusted sensors.

5.1 Anonymity

As discussed in Section 2, all TPMs support anonymous attestations, in which an attestation identity key (AIK) is obtained from a privacy CA. If a sensor reading is signed with an AIK, an application can verify that the reading has been signed by a valid TPM without revealing which TPM signed it. This mechanism can easily be adopted for trusted sensors. One drawback of the current anonymous attestation scheme is that the identity of the signing TPM is revealed to the third-party privacy CA. However, with version 1.2, TPMs use direct anonymous attestations, a protocol that eliminates the need for a privacy CA [3].

5.2 Non-Transferability

To ensure that sensor readings are non-transferable, a user cannot transmit the signed reading to an application. If the user did, the application could hold on to the signed reading and reuse it later: the reading is verifiable by anyone because of the digital signature. Thus, the user must convince the application of the sensor reading's validity *without* having to transmit the sensor reading's signature.

In cryptography, such problems are typically solved with zero-knowledge protocols [10] that provide the following privacy guarantee: one party can prove to another that a statement is true without revealing anything beyond the veracity of the statement. For trusted sensors, a zero-knowledge protocol for verifying the sensor signatures would reveal the sensor reading but not the actual signature. Unfortunately, there are no known efficient zero-knowledge protocols that use standard RSA-based cryptography. A practical alternative to zero-knowledge protocols are witness hiding proof of knowledge (WHPOK) protocols. WHPOK protocols are relaxations of zero-knowledge protocols that meet the desired non-transferability requirement [6]. In contrast with zero-knowledge which guarantees that an application will not learn *any* information beyond the validity of the signature, WHPOK only guarantees that the application does not receive a copy of the signature, and is un-

able to learn how to prove the signature's validity. A mathematical description of WHPOK properties can be found in [6].

It is possible to efficiently convert a traditional RSA signature into a WHPOK protocol [11]. This makes WHPOK relatively easy to implement because it relies on traditional RSA for which there are well-known cryptographic libraries available. In previous work, we implemented a WHPOK protocol; a more rigorous description of our implementation and an evaluation can be found in [24].

6. RELATED WORK

Another project has independently proposed using trusted sensors for mobile sensing [9]. Their work is similar to our first design; it assumes the presence of a secure environment based on virtualization as well as a TPM that is used to generate attestations. A remote service uses the attestations to verify that the readings returned by the mobile device are authentic. Our work provides an additional approach to building trusted sensors with increased security and privacy properties; our second design has stronger security privacy than [9] (or our design #1), and we also present a cryptography mechanism based on zero-knowledge protocols that increases users' privacy.

A few previous research projects have proposed using trusted computing for location-based applications. In [5], the authors build a trusted sensing peripheral that incorporates a TPM and a GPS sensor to reduce the possibility of data pollution in participatory sensing applications. The focus of this previous work is on the custom-made implementation of a trusted GPS board and its performance; instead, our focus is more broad on making any sensors trusted, and on dealing with privacy issues. Another relevant though less related project is [14], in which users can verify that a server does not compromise their privacy by relying on secure logging and trusted computing.

Previous work has studied reputation management and voting schemes to eliminate corrupted data readings from participatory sensing applications [8, 7, 17]. Although trusted sensors make it much harder to mount attacks against participatory sensing applications, some of these applications collect information supplied by users, rather than sensors. For these applications, reputation management and voting schemes are likely to continue to be important security tools.

7. CONCLUSIONS

In this paper, we present a diverse set of mobile applications that can be enabled by including trusted sensors on mobile devices. We describe two different designs for making sensor readings trusted. The first design relies on a TPM and on a virtualized environment to provide trusted sensor readings. The second design is based on incorporating trusted computing primitives directly into sensors. While the first design is less secure, being susceptible to hardware attacks, it also has a lower barrier to deployment than the second design. Finally, we discuss the privacy issues arising from the use of trusted sensors and how anonymous credential schemes, zero-knowledge protocols, and witness-hiding protocols can overcome them.

8. REFERENCES

[1] AMD. AMD64 Architecture Programmer's Manual, Volume 2, 2009. http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/24593.pdf.

[2] App Games. Swine Flu Tracker Map iPhone Game, 2009. http://www.1888freeonlinegames.com/iphonegames/free-swine-flu-tracker-map-119.html.

[3] E. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation. In *Proc. of 11th CCS*, October 2004.

[4] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless Device Identification using Radiometric Signatures. In *Proc. of 14th Mobicom*, September 2008.

[5] A. Dua, N. Bulusu, and W. chang Feng. Towards Trustworthy Participatory Sensing. In *Proc. of the 4th HotSec*, August 2009.

[6] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of 22nd STOC*, Baltimore, MD, May 1990.

[7] C. Fretzagias and M. Papadopouli. Cooperative location-sensing for wireless networks. In *Proceedings of the 2nd PerCom*, Orlando, FL, March 2004.

[8] S. Ganeriwal and M. B. Srivastava. Reputation-based framework for high integrity sensor networks. In *Proceedings of the ACM Workshop on Security in Ad-hoc & Sensor Networks (SASN)*, Washington, DC, October 2004.

[9] P. Gilber, L. P. Cox, J. Jung, and D. Wetherall. Toward trustworhy participatory sensing. In *Proceedings of the 11th HotMobile*, Annapolis, MD, February 2010.

[10] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the 17th STOC*, Providence, RI, May 1985.

[11] S. Goldwasser and E. Waisbard. Efficient transformation of well known signature schemes into designated confirmer signature schemes. *TR: MCS03-13, The Weizmann Institute of Science*, 2003.

[12] Google. Panoramio – Photos of the World, 2009. http://www.panoramio.com.

[13] R. Gummadi, H. Balakrishnan, P. Maniatis, and S. Ratnasamy. Not-a-Bot (NAB): Improving Service Availability in the Face of Botnet Attacks. In *Proc. of the 6th NSDI*, April 2009.

[14] U. Hengartner. location privacy based on trusted computing and secure logging. In *Proc. of the 4th SecureComm*, September 2008.

[15] Intel. Intel Trusted Execution Technology, 2009. http://download.intel.com/technology/security/downloads/315168.pdf.

[16] V. Lenders, E. Koukoumidis, P. Zhang, and M. Martonosi. Location-based Trust for Mobile User-generated Content: Applications, Challenges and Implementations. In *Proc. of the 9th HotMobile*, February 2008.

[17] D. Liu, P. Ning, and W. Du. Attack-resistant location estimation in sensor networks. In *Proceedings of the 4th IPSN*, Los Angeles, CA, April 2005.

[18] P. Mohan, V. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proc. of the 6th SenSys*, November 2008.

[19] E. Paulos, R. Honicky, and E. Goodman. Sensing Atmosphere. In *Proc. of the Workshop on Everyday Mobile Phones in Support of Participatory Research*, November 2007.

[20] Purdue University. Cell phone sensors detect radiation to thwart nuclear terrorism, 2008. http://www.purdue.edu/UNS/x/2008a/080122FischbachNuclear.html.

[21] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *Proc. of 10th HotMobile*, February 2009.

[22] SKYHOOK Wireless. Skyhook, 2009. http://www.skyhookwireless.com.

[23] N. O. Tippenhauer, K. Rasmussen, C. Popper, and S. Capkun. Attacks on public wlan-based positioning systems. In *Proc. of the 7th MobiSys*, June 2009.

[24] A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman. Lockr: Better Privacy for Social Networks. In *Proc. of the 5th coNEXT*, December 2009.

[25] Trusted Computing Group. TMP Specification version 1.2, 2009. www.trustedcomputinggroup.org/specs/TPM/.

[26] Wi-Fi FreeSpot. Directory, 2009. http://www.wififreespot.com.