Trusted Sensors for Participatory Sensing and IoT Applications based on Physically Unclonable Functions

Ihtesham Haider
Institute of Networked and Embedded Systems
Alpen-Adria-Universität
Klagenfurt, Austria
ihtesham.haider@aau.at

Michael Höberl
Alpen-Adria-Universität
Klagenfurt, Austria
Technikon Forschungs- und Planungsgesellschaft mbH
hoeberl@technikon.com

Bernhard Rinner
Institute of Networked and Embedded Systems
Alpen-Adria-Universität
Klagenfurt, Austria
bernhard.rinner@aau.at

ABSTRACT

With the emergence of the internet of things (IoT) and participatory sensing (PS) paradigms trustworthiness of remotely sensed data has become a vital research question. In this work, we present the design of a trusted sensor, which uses physically unclonable functions (PUFs) as anchor to ensure integrity, authenticity and non-repudiation guarantees on the sensed data. We propose trusted sensors for mobile devices to address the problem of potential manipulation of mobile sensors' readings by exploiting vulnerabilities of mobile device OS in participatory sensing for IoT applications. Preliminary results from our implementation of trusted visual sensor node show that the proposed security solution can be realized without consuming significant amount of resources of the sensor node.

Keywords

PUF; trusted sensors; internet of things; participatory sensing; mobile applications; trust; privacy

1. INTRODUCTION

The internet of things (IoT) is envisioned to be a network of pervasively present physical objects, embedded with sensors and actuators. These objects can interact and cooperate with each other through unique addressing to create smarter environments/spaces permeating in domains such as transportation, cities, health-care, energy, tourism, and industry [29]. Sensors have an essential role in this emerging new paradigm. Participatory sensing (PS) is an interesting sensing approach that aims to include the smartphones in the sensing loop and utilize them as sensory stations taking a multi-sensor snapshot of their immediate environment. By intelligently combining these individual sensor readings it is possible to create a clear picture of the physical world that can be used as an input to various IoT applications such as urban sensing [3], health-care [1] and citizen journalism [2] etc.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

IoTPTS'16, May 30-June 30 2016, Xi'an, China

© 2016 ACM. ISBN 978-1-4503-4283-4/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2899007.2899010

The scale and ubiquitous nature of this sensing paradigm present a multitude of challenges: First, PS is open to public participation. Sensor readings obtained with mobile device sensors carry no security guarantees. The lack of integrity and authenticity guarantees on mobile sensor readings [22] and security flaws in mobile OS (e.g., *Fake ID* [24], and *Master Key* [25]) allow malware, such as worms, to launch *data pollution* attacks on a central server database by uploading manipulated or fabricated sensor readings [22, 8]. Malicious users can submit false readings to increase their reward which could be either monetary, reputation, or service quota. Preventing these data pollution attacks is a major challenge in participatory sensing. Second, one approach to counter these attacks is to provide integrity and authenticity guarantees on the sensor readings right from the onset; however, any sensor-centric solution for making sensor readings trustworthy must be lightweight for the resource-constrained sensors. Third, a variety of sensors with different amount of resources and processing capabilities have found their way into the mobile devices. Therefore presenting a generic solution for a wide spectrum of sensors is another challenge.

In this paper, we present the design of trusted sensors, which offer integrity, authenticity and non-repudiation guarantees on the sensed data by leveraging PUFs as root of trust. Sensed data integrity, authenticity and non-repudiation is ensured by having the sensor firmware sign the data, and integrity of the firmware is achieved by secure boot of the sensor. The on-chip PUF serves as secure key storage for digital signatures and secure boot processes. The security of the design relies on proven properties of the underlying primitives, i.e., physical unclonability and tamper-evidence of PUFs, and existential unforgeability of digital signatures. Current approaches such as trusted platform module (TPM) based trusted sensors propose the integration of TPM into the sensor nodes. This requires extensive hardware modifications to the resource constrained sensors which might be uneconomical for low-end sensor nodes. This PUF-based design maps onto a typical low-end sensor hardware and does not require any hardware modifications. Although, we are primarily focused on mobile sensors, this design is also applicable to standalone sensors for IoT applications.

The remainder of this paper is organized as follows. Section 2 describes the system and threat models. Section 3 discusses previous work in areas of physically unclonable functions and trusted sensors, and Section 4 introduces building blocks and presents architecture of the proposed trusted sensor. Section 5 addresses various design challenges to enable

trustworthy participatory sensing by incorporating the proposed trusted sensors in mobile devices. It also presents a road-map for privacy-aware trustworthy participatory sensing. Section 6 discusses implementation status of the proposed work and presents preliminary results and Section 7 concludes this paper.

2. SECURITY MODEL

In this section, we describe the architecture of participatory sensing system, the threat model, the security goals that we aim to achieve, and our trust assumptions.

2.1 System Model

Entities that participate in a typical participatory sensing task are a querier (Q), a participatory sensing server (PSS), and mobile nodes (MNs). In a given participatory sensing application, Q is an end-user interested in receiving the sensed data. It is also referred to as client of the application. PSS is a central entity in the participatory sensing infrastructure that governs a sensing task. It is responsible for receiving queries from Q, creating new sensing tasks, collecting sensed reports from the participating MNs, extracting meaningful information from the sensed data and providing Q with this information. MNs are the mobile devices, equipped with a multitude of sensors, that can contribute sensed data to a sensing task. A client application (App), installed on the MN, reads the mobile sensors using underlying operating system APIs and forwards the sensed reports to the PSS. A generic sensor node architecture is illustrated in Figure 1. MNs can leverage Wi-Fi or a cellular network service to submit their reports to the PSS. Report is a data structure which contains sensed data and associated metadata such as location of the sensor and time-stamp of sensed reading.

2.2 Threat Model

In this work, we address attacks to manipulate or fabricate sensed data by compromising mobile OS. We consider malicious mobile device custodians or malicious third party as adversary. The adversary is assumed to be able to compromise the OS on the host device. Our goal is to ensure authenticity of sensing hardware as well as integrity and non-repudiation of sensed readings to remote server. Attacks based on the internet and sensor networks such as denial of service or selective forwarding are not considered in this work. We recommend implementation of PUF on the sensor controller. However, if sensing unit is leveraged for PUF instantiation, we assume that the communication between the sensing unit and the controller cannot be compromised by the adversary. When used as PUF, the sensor SRAM should be initialized immediately after it has been read during the sensor boot-up. This renders the SRAM PUF unavailable for the adversary. The client application, App, running on the mobile device is assumed to be trusted by the PSS and is verifiable through remote attestation.

3. RELATED WORK

3.1 Physically Unclonable Functions (PUFs)

PUFs are lightweight, energy efficient hardware security primitives which typically exhibit a challenge-response behavior. When queried with a challenge $\mathbf{c}$, the PUF generates a response $\mathbf{r}$, that depends on $\mathbf{c}$ and the uncontrollable CMOS manufacturing details of the underlying hardware. Since these manufacturing variations are random and unique for every PUF instance, the corresponding PUF output is also random and unique. Randomness, uniqueness, unclonability, and tamper resistance make PUFs interesting candidates for cryptographic applications such as key generation and authentication.

	Gen: $(W, K) \leftarrow \texttt{Gen}(\mathbf{r})$	**Rep:** $K \leftarrow \texttt{Rep}(\mathbf{r}', W)$
Code-Offset	$C_s \leftarrow \texttt{Encoding}(\mathbf{x})$, where $\mathbf{x}$ = random and $C_s \in \mathcal{C}$ $W \leftarrow \mathbf{r} \oplus C_s$ $K \leftarrow \texttt{Hash}(\mathbf{r})$	$C_s' \leftarrow \mathbf{r}' \oplus W$ $C_s \leftarrow \texttt{Decoding}(C_s')$, if $\texttt{Hamming distance}(C_s, C_s') \leq t$ $\mathbf{r} \leftarrow C_s \oplus W$ $K \leftarrow \texttt{Hash}(\mathbf{r})$
Syndrome	$W \leftarrow \mathbf{r} \cdot \mathbf{H^T}$, where $\mathbf{H^T}$ is parity check matrix of selected linear block code $K \leftarrow \texttt{Hash}(\mathbf{r})$	$\mathbf{s} \leftarrow \mathbf{r}' \cdot \mathbf{H^T}$, where $\mathbf{s}$ is syndrome Determine $\mathbf{e}$ from: $\mathbf{r}' \cdot \mathbf{H^T} \oplus W = \mathbf{e} \cdot \mathbf{H^T}$, where $\mathbf{e}$ is the error vector $\mathbf{r} \leftarrow \mathbf{r}' \oplus \mathbf{e}$ $K \leftarrow \texttt{Hash}(\mathbf{r})$

Table 1: Code-offset and syndrome constructions.

A PUF's challenge-response behavior does not correspond to a random oracle as a PUF response is not perfectly reproducible and is non-uniformly distributed. Multiple responses obtained from the same PUF slightly differ from one another. A commonly used measure to quantify this error (also called noise) is the Hamming distance. The Hamming weight is used to measure the deviation of a PUF response from a uniform distribution. A PUF-enabled device is authenticated if the Hamming distance between a regenerated response and the one recorded during manufacturing is negligible. When used as a secret key, a PUF response needs to be perfectly reproducible and uniformly distributed. To make the PUF response reproducible and uniformly distributed a helper-data algorithm (HDA), that features a fuzzy-extractor or a secure-sketch scheme [7], is required. The HDA consists of two stages: information reconciliation which deals with noise by using error-correction codes and privacy amplification which uses a hash function to ensure a nearly uniform PUF response.

A PUF-based key is generated in two phases: (i) key enrollment and (ii) key regeneration. Enrollment is a one-time process, performed during manufacturing in a secure and controlled environment. During this phase, based on the PUF response $\mathbf{r}$, helper data W and a key K are generated. Helper data is integrity protected, publicly stored information. Later in the field, the PUF is re-evaluated resulting in a slightly noisy response $\mathbf{r}'$. W and $\mathbf{r}'$ are fed as input to the regeneration phase where error correction of $\mathbf{r}'$ is performed to obtain $\mathbf{r}$ (if $\texttt{Hamming distance}(\mathbf{r}, \mathbf{r}') \leq t$, where t is error correction capacity of the selected code $\mathcal{C}$). This reproduced PUF output is fed to the privacy amplification stage, commonly employing a universal hash function, which results in a fixed length bit string to be used as a key or a random seed for key-pair generation. Two constructions of fuzzy extractors namely *Code-Offset* and *Syndrome Construction* dominate the HDA implementation landscape. In a fuzzy extractor, enrollment and reconstruction phases are called **Gen** and **Rep** respectively as illustrated in Table 1.

A number of PUF circuits have been reported so far. Based on their working principle [18], PUFs are often categorized into *delay-based* PUFs e.g., arbiter PUF [17] and ring oscillator PUF [11] and *memory-based* PUFs e.g., SRAM

PUF [14]. The former approach exploits the delay difference experienced by signals on two equal paths whereas the latter approach exploits the random start-up values of SRAM memory cells. Based on their available challenge space, PUFs are categorized as either *weak* or *strong*. Weak PUFs offer only a few challenge-response pairs (CRPs) which scale linearly with the required logic area on an IC. They are used for secret key generation. A popular example is the SRAM PUF. Strong PUFs offer a huge number of CRPs which often scale exponentially with the required logic area. They exceed the requirements for key generation and are promoted for lightweight authentication without cryptographic primitives. The arbiter PUF is a prominent example for a strong PUF.

3.2 Trusted Sensors

Early work on trusted sensors [8] was motivated by participatory sensing. Trusted computing hardware such as trusted platform module (TPM) was proposed for mobile devices to attest the sensed data. Saroiu and Wolman [22] proposed the integration of TPM into the mobile device sensors which may not be an economical solution for resource constrained embedded applications. Moreover, TPMs are vulnerable to physical attacks. Winkler et al. [30] introduced TPM to secure embedded camera nodes. Potkonjak et al. [20] proposed an alternative approach for the trusted flow of information in remote sensing scenarios that employed public physically unclonable functions (PPUFs). PPUFs are hardware security primitives which can be modeled by algorithms of high complexity. As a result, PPUFs enable public key cryptography without using cryptographic primitives. For any given input, the PPUF hardware (secret key) output is many orders faster than its software counterpart (public key). However, current PPUF designs involve complex circuits requiring high measurement accuracy which slows down the authentication process and therefore it is not a scalable solution. Rosenfeld et al. [21] presented the idea of a sensor PUF, whereby the PUF response depends on the applied challenge as well as the sensor reading. In a recent work, Cao et al. [5] proposed a CMOS image sensor based weak PUF, which generates response bits by exploiting the random fixed pattern noise in a selected pixel pairs.

Our approach uses on-chip PUF to serve as identity of the sensor and provide secure keys for ensuring non-repudiation of the sensed data as well as integrity of the sensor firmware. With reference to TPM-based solutions, this PUF-based design is lightweight, has better physical security properties, and requires no modifications in the sensor hardware. Compared to PPUF based approach, PUFs offer simpler, faster and scalable solution.

4. PUF-BASED TRUSTED SENSOR

A typical sensor node comprises a sensing unit, an analog to digital converter (ADC), a controller, a power source and an external memory as illustrated in Figure 1. The controller controls the functionality of the other components of the sensor node and performs data processing. The most common choice for the controller is a lightweight microcontroller (MCU) due to its low cost, ease of programming and low power consumption. Alternatives include System on Chip (SoC) and FPGAs for applications such as real-time video processing for image sensors. In this section, we discuss three security features: *PUF-based Secure Key Generation and Storage*, *Sensed Data Attestation*, and *Secure Boot of Sensor Controller* which serve as the building blocks of the trusted sensor design and can be realized using existing resources on a typical sensor node (Figure 1). *PUF-based Secure Key Generation and Storage* module provides secure keys to (i) *Sensed Data Attestation* to ensure integrity and non-repudiation of sensed data as it leaves the trusted sensor and (ii) *Secure Boot of Sensor Controller* to ensure integrity of the sensor platform at every boot-up.

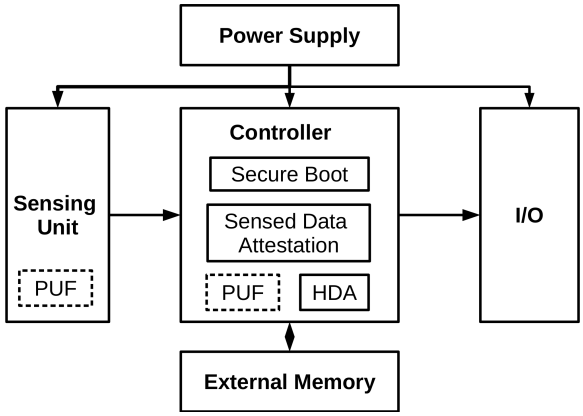

Figure 1: Overview of PUF-based trusted sensor design.

4.1 PUF-based Key Generation and Storage

We propose PUF-based secure key generation and storage on the resource constrained trusted sensor nodes, by leveraging the framework [13] we proposed earlier. A PUF-based key generation approach offers (i) a unique key for each sensor based on the intrinsic properties of sensor hardware, (ii) a low-cost, secure storage of the key, (iii) a binding of the key to the node, and (iv) the availability of the key with negligible probability of failure. For details on the PUF-based key generation process see Section 3.1. The architecture of the PUF-based key generation and storage module has two cascaded components: a PUF source and the helper data algorithm (HDA). The design of HDA depends on the choice of the PUF source.

4.1.1 PUF source

Various PUF sources are either inherent to a typical sensor node or could be implemented using existing sensor resources: (i) Uninitialized on-chip SRAM, available on most lightweight MCUs as well as SoCs, have PUF properties and can be used as SRAM PUF [28]. (ii) If SRAM is either not available or gets initialized at the start up, a ring-oscillator (RO) PUF can be implemented in ASIC (sensing unit) or FPGA section of SoC (controller). However, RO PUFs add some hardware overhead. (iii) Various sensor specific PUFs [5, 21] have been identified which exploit the manufacturing variations in the sensing circuitry. For instance, CMOS image sensor based PUF [5] exploits the random fixed pattern noise in a selected pixel pairs. (iv) The calibration errors of mobile sensors such as accelerometer and microphone have also potential to uniquely identify mobile device [4]. However, whether these identifiers carry enough entropy to be used as cryptographic keys (PUFs) is not yet established and is an active research topic.

4.1.2 Helper Data Algorithm

The Helper Data Algorithm (HDA) (see Section 3.1) ensures (i) a perfectly reproducible PUF response under a range of environmental and operating conditions, i.e., the Hamming distance between any two responses from the same PUF is zero, and (ii) an almost uniformly distributed PUF response, i.e., the Hamming weight of any PUF response is around 50%. The helper data algorithm is designed based on the *error-rate* (the Hamming distance between two responses obtained from the same PUF instance over worst-case environmental and voltage changes) and the *entropy* of the PUF source. Guajardo et al. [12] showed that SRAM PUFs have a secrecy-rate of 0.76 bits/SRAM bit. Independent response bits from RO PUF also have a high entropy. The Hamming distance and Hamming weight of PUF responses follow a binomial distribution [27]. Assuming PUF response bits to be independent, the probability that an n bit regenerated PUF response will have more than t errors, denoted by P_{Fail}, is given by [12]

$$P_{\text{Fail}} = \sum_{i=t+1}^{n} \binom{n}{i} {p_b}^i (1-p_b)^{n-i} = 1 - \sum_{i=0}^{t} \binom{n}{i} {p_b}^i (1-p_b)^{n-i} \quad (1)$$

where p_{b} is the bit error probability (*error-rate*) of PUF response bits. P_{Fail} is desired to be negligible, i.e., $\leq 10^{-6}$. Based on the desired values of t, n, P_{Fail}, and the nature of errors, the error correcting code is selected. The selected code determines the number of source bits, i.e., the number of PUF response bits required to obtain the desired number of error free bits.

Error correction results in entropy loss which corresponds to the maximum number of helper data bits λ exposed. For instance, let us assume an $\{n, k, t\}$ linear block code in the code-offset or syndrome construction. The PUF generates n bits, and during HDA phase $n-k$ bits of helper data (public information) is produced. So the entropy loss is given as $\lambda = n - k$. If n' is the minimum entropy of the n-bit PUF response, with a λ bit loss, the residual entropy $n' - \lambda$ should be sufficient for required key. Hashing compresses the error corrected response into a smaller bit string with the same entropy thereby increasing the entropy per response bit.

4.2 Sensed Data Attestation

The goal of sensed data attestation is to prevent any manipulation of the trusted sensor readings by the mobile OS. Using the PUF response as a source of randomness, a public-private key pair, so-called *attestation key*, is generated and the sensed data is signed with the private half of the *attestation key* before it leaves the sensor. Therefore, integrity and authenticity of the sensed data are ensured independent of the state of the mobile OS and can be verified by a remote entity, e.g., by verifying the signatures on the sensed data. Attestation can be realized using the Elliptic Curve Digital Signature Algorithm (ECDSA) as the hardness of elliptic curve discrete logarithm problem (ECDLP) offers much smaller sizes for domain and key parameters compared to RSA and DL based schemes. This in turn affects the performance (i.e., computation and energy consumption) of cryptographic algorithms based on these schemes. Data attestation can be implemented in the firmware of the sensor controller.

4.3 Secure Boot of Sensor Controller

Secure boot initializes an embedded system with a trustworthy configuration. Secure boot, using PUF generated key, can bind the trustworthy firmware with the PUF-enabled hardware. PUF-based secure boot of the sensor node controller ensures trustworthiness of the firmware at every boot up thereby providing deterrence against the attacks aimed at manipulating the sensor firmware. When an embedded system is initialized, boot-loader is the first code that executes before the main program (i.e., firmware or OS kernel) and is responsible for initializing the necessary hardware, loading the firmware, and passing on the system control to the firmware.

Secure boot requires the boot process to start with a trusted, immutable code and every phase of boot process must verify the integrity of next phase before passing on the system control to the next phase. To meet the first requirement, the first piece of boot code can be placed in immutable memory, written to only by the manufacturer. For low-end MCUs, the boot-loader is a small piece of code and can fit into on-chip immutable memory such as a masked ROM. For complex SoCs, the boot-loader is large and is executed in multiple stages. The first stage boot-loader (FSBL) is generally smaller in size and can reside in the immutable ROM. Other stages of the boot-loader, the firmware and the OS kernel reside in the external flash memory as illustrated in Figure 2. The integrity of every stage of boot process can be verified either (i) using symmetric key techniques i.e., by verifying the hash and decrypting the previously encrypted next phase boot code or (ii) using asymmetric key techniques i.e., by verifying signatures of the next phase boot code. If the verification succeeds the system control is transferred to next executable code, otherwise the boot process is aborted. Start of the boot process with the immutable code and validation of each next phase by the previous phase of the boot process ensures a chain-of-trust all the way from FSBL to firmware, OS kernel or applications. Using the PUF-based key for boot code validation additionally binds the trusted firmware to the sensor hardware.

PUF-based secure boot is implemented in two phases: enrollment and regeneration. Enrollment is one time process, performed during manufacturing, during which PUF-based key and helper data are generated by Phase 0 boot code and subsequent Phase 1 code e.g., SSBL or firmware is either signed or encrypted-then-hashed using PUF-based key. For multi-stage boot process, PUF response could be used as secure seed for PRNG to generate separate keys for validating every phase of boot process. Hash or signature values calculated during enrollment and the verification algorithms are also saved in the immutable ROM to be used in regeneration phase. Regeneration is performed at every boot-up. During Phase 0 code execution from the masked ROM, PUF-based key is regenerated using the helper data saved during enrollment. Before passing on the system control to Phase 1, it is loaded from the flash, decrypted and its integrity is validated using the reference hash or signature values present in the immutable ROM. This chain of verification at every stage of boot process leads to initialization of MCU or SoC in known, trusted configuration as illustrated in Figure 2. Schaller et al. [23] recently proposed an SRAM PUF based anti-counterfeit solution that binds the firmware with the hardware.

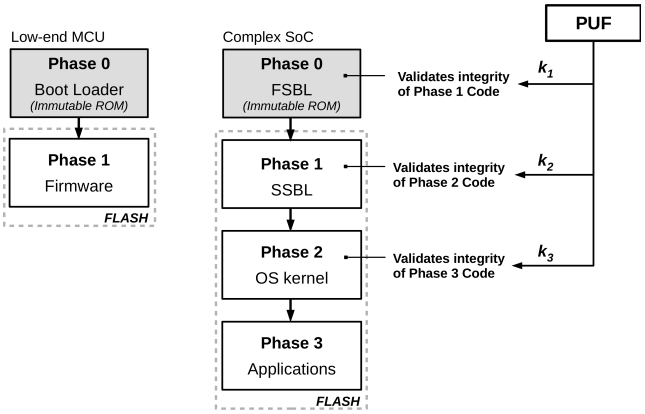

Figure 2: Overview of PUF-based secure-boot process for low-end MCU based low-end devices (e.g., sensor nodes) and SoCs based complex embedded systems (e.g., mobile devices).

4.4 Trusted Sensor Architecture

Figure 1 illustrates how the proposed features of a trusted sensor can be mapped onto a typical low-end sensor node without any significant hardware modifications. The choice of the PUF source determines whether it can be realized on the sensing unit (sensor PUFs) or the sensor controller (SRAM PUF, RO PUF etc.) Sensed data attestation is implemented in the firmware of the sensor controller, which receives the sensed data from the sensing unit, signs it and sends it to output interface of the sensor node. Secure boot is achieved by modifying the boot-loader of the sensor controller. This PUF-based approach provides integrity, authenticity and non-repudiation on sensed data right from the onset. Hardware state of the sensor is protected by the tamper resistance of the on-chip PUF. Trustworthiness of the sensor firmware is ensured by secure boot. If incorporated in the mobile devices, these trusted sensors can provide protection against attacks aimed at manipulating the sensors' readings at mobile OS level as sensed data leaves the trusted sensor along with signatures.

4.5 Security Discussion

The integrity and non-repudiation of sensed data is ensured by having sensor firmware sign the data. Manipulating a mobile device based trusted sensors' readings by exploiting the mobile software stack requires existential forgery of ECDSA, the employed digital signature scheme. However, ECDSA has been proven existentially unforgeable against adaptive chosen-message attacks under generic group and collision-resistant hash assumptions. Integrity of the sensor firmware is verified during every boot-up. If any manipulation in the firmware is detected, the boot process is aborted and respective error code is returned to the client application. PUF modeling attacks are not relevant for weak PUFs however, other attacks can be realized with invasive or non-invasive techniques. PUFs are tamper-proof which makes the trusted sensor robust against the invasive attack techniques. The non-invasive techniques i.e. side-channel attacks on PUF can be deterred by breaking the correlation between the leaked information and the circuit operation e.g. by reduction/elimination of the leaked information. However, we did not implement these countermeasures in our implementation. During the enrollment phase, an additional hash value of the helper data can detect any manipulation in the helper data later. If an additional link-list containing RO pairs has to be stored as well, it has to be included in the hash as well. Another potential point of attack is error correction code if used in a code-offset construction. Codeword-masking [20] uses the linearity of the used code as a countermeasure and provides deterrence against differential power analysis (DPA) attacks.

5. TRUSTWORTHY PARTICIPATORY SENSING WITH TRUSTED SENSORS

As described in Section 2.1, the main entities involved in a participatory sensing task are the sensors embedded in the mobile hardware, a client application (App) that runs on the mobile device, and a remote server (PSS). To ensure integrity of the sensed data right from the onset, we propose the incorporation of the PUF-based trusted sensors in the mobile devices. However, in order to ensure trustworthy participatory sensing few design challenges need to be addressed.

First, an individual trusted sensor (other than GPS) does not provide any guarantees on time and location of the sensed data. Such knowledge is crucial since standalone sensor readings are much less useful than sensor readings with time and location information. Today almost every mobile device is equipped with a GPS receiver which allows it to accurately determine its position and time. Signed location and time values from the trusted GPS (proposed design integrated into the GPS module) can provide proof of location and time.

Second, how to bind the readings from multiple trusted sensors (e.g. camera and GPS)? We suggest that the client App provides a nonce to the trusted camera and the GPS while requesting the sensed data. Each sensor appends the nonce to its freshly sensed data and signs it. The client application verifies the received sensed data from the camera and the GPS using the nonce and binds the data from two sensors upon successful verification. This simple approach ensures fresh data from multiple trusted sensors. However, the underlying assumption in this approach is that the delay between multiple sensors' responses, especially between GPS and other sensors, does not exceed a threshold. The value of this threshold is application specific and is left to the application developers. Another assumption in this approach is that the client application (App) running on the mobile device, performs its functionality as assigned by the remote server, PSS, i.e., in trustworthy manner. Remote software attestation [26, 10, 15, 16] provides the mechanism whereby the integrity of code running on an untrusted hardware can be remotely verified.

The mobile device registers with the PSS by downloading the client application (App). The PUF-enabled mobile sensors are authenticated by App on providing proof of knowledge of PUF-response to randomly selected challenges. Once authenticated, trusted sensors output signed data to the App. The App can verify the signatures on individual sensor readings, strip the signatures, bind multiple readings together and forward this data to the server in secure manner either using encrypt-then-MAC or signcryption.

5.1 Towards Privacy-Aware Trustworthy Participatory Sensing

Successive submission of signed sensed data can however reveal sensitive personal information about the mobile device custodian to the PSS, for instance, location trace of the user can reveal frequently visited places such as home, work, religious affiliations, and entertainment interests etc. The user privacy can be protected by ensuring unlinkability between successively submitted sensor readings by a user. We foresee a pseudonym based scheme as a potential solution for user privacy. A pseudonym is a unique identifier that allows authentication of pseudonym holder without knowledge of his real identity. A pseudonym can offer pseudonymity as all actions authenticated with this pseudonym are linkable to each other but not to pseudonym holder's real identity. Pseudonymity in participatory sensing is vulnerable to inference based attacks. However, if an entity holds a set of pseudonyms and changes its pseudonym over time, the actions performed with different pseudonyms could neither be linked to each other nor to the real identity of these pseudonym holder thereby providing unlinkability.

One way to achieve this is by having a strong-PUF based pseudonym generator on the mobile device. This scheme requires a trusted third party (TTP) in the infrastructure such as a pseudonym registration authority (PRA). To generate a pseudonym, a random challenge is given to the strong-PUF. Using the PUF response, a public-private key pair is generated which serves as a pseudonym. Pseudonym acts as a short-term identifier of the mobile device for submitting sensor readings to the PSS. Before a mobile can use a pseudonym to contribute sensed data to the application server, the pseudonym needs to get registered with the PRA, which keeps track of the mapping between the long-term identity and the currently active pseudonym of the mobile device. Upon successful registration, the PRA certifies the active pseudonym of the mobile device. The registration of a pseudonym with the PRA ensures that at any given time a registered mobile device has only one active pseudonym. Using the active pseudonym and the certificate from PRA, the mobile device can register and contribute sensed data to the PSS. After a random time interval, a pseudonym change process is initiated by the mobile device. During pseudonym change, a new pseudonym is generated and is registered with the PRA. Upon successful pseudonym change, the PRA updates the active pseudonym in its database. PRA can communicate the list of inactive/revoked pseudonyms to the PSS to prevent Sybil attacks. For effectiveness, pseudonym change must encompass all network layers [9] to prevent any trivial linking between the pseudonyms [19] using IP or MAC address. Moreover, to make pseudonym change effective it is recommended that multiple mobile devices change their pseudonyms simultaneously. Identification of such situations is out of the scope of this work.

6. TRUSTED SENSOR IMPLEMENTATION

In this section, we present preliminary results from our implementation of a PUF-based key generation module for the prototype trusted visual sensor node (VSN) shown in Figure 3. The prototype VSN uses an Omnivision's 5642 image sensor and Xilinx's Zynq7010 SoC as the sensor controller. The SoC is comprised of reprogrammable logic equivalent to $\approx 430k$ ASIC gates and dual ARM Cortex-A9 cores. The

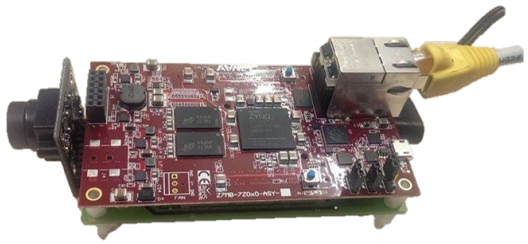

Figure 3: Trusted camera prototype

basic sensor readout functionality is implemented on the SoC as depicted in Figure 4. In summary, the ARM Cortex-A9 configures the image sensor for 640x480 pixels, YUV 4:2:2 color space and 15fps. The *Format Adapter* reads the image sensor using camera parallel interface and converts it to an AXI streaming interface. *DMA* uses triple buffers to temporarily store the video data in external SDRAM. The processor reads the stored data, performs basic image processing and enhancement algorithms e.g., edges enhancement and transmits this video frames to a host PC using an Ethernet connection. This prototype is realized as a modular design comprised of three modules: *PUF based key generation*, *sensed data attestation*, and *secure boot of sensor controller* to be implemented on the SoC. Initial results from the *PUF based key generation and storage* module are discussed here, where as implementation of rest of the modules is ongoing work.

6.1 PUF-based Key Generation Module

The goal of this module is to generate a 128 bit key with high entropy. On Zynq7010 SoC, the available on-chip SRAM gets initialized at the start up. Once initialized SRAM cells cannot be used as PUF. Therefore, we implemented a ring oscillator (RO) PUF. A total of 1344 3-stage ROs were implemented using reprogrammable logic-area of SoC by exploiting the *Hard Macro* utility in Xilinx ISE, which assigns the same routing to each ring oscillator. Two ROs are selected using multiplexers and the frequencies of selected ROs are quantified by using two counters. In order to achieve a fair comparison, a third reference counter uses the board's clock as an input and triggers the selected ROs and their frequency measuring counters on and off synchronously. When the reference counter reaches a predefined value, it stops both counters. The values of the two counters are compared and the result is stored in a register. The comparison of two counter values yields to a response bit. However, each RO frequency change with varying environmental and operating conditions. And more importantly, the frequency of each RO changes at a different rate. So, if two ROs being compared have slightly different frequencies and their frequencies vary at different rates then there is likelihood of a bit-flip in the PUF response with change in temperature or voltage. Hence, choosing RO pairs that produce reliable response bits can save considerable effort in the helper data algorithm (HDA) phase. The simplest approach, called Chain of Neighbors (CoN) [6], is to compare neighboring ROs, as they are expected to be exposed to similar environmental conditions. Sequential Pairing Algorithm (SPA) [31] pairs ROs with frequency difference greater than threshold. Both approaches can produce up to $\lfloor N/2 \rfloor$ reli-

	CoN Pairing	SPA Pairing
Intra-Hamming Distance	7%	1.4%
Inter-Hamming Distance	49%	49%
Hamming Weight	49.3%	51.2%

Table 2: Noise, uniqueness and uniform distribution of RO PUFs based on two RO pairing strategies i.e. CoN and SPA.

Code (n, k, d)	P_{Fail}	# of source bits
RM (16,5,8)	$2.401331 * 10^{-3}$	410
Rep. (9,1,9)	$3.770032 * 10^{-7}$	1152
BCH (31,6,15)	$1.338228 * 10^{-7}$	662

Table 3: Comparison of error correcting codes for 2% bit error-rate.

able bits from N ROs. In our implementation, 672 bits were generated from 1344 ROs. Table 2 summarizes the noise (intra-Hamming distance), uniqueness (inter-Hamming distance) and uniform distribution (Hamming weight) of the implemented RO PUF. These parameters are reported for the two RO pairing strategies described above: Chain of Neighbors (CoN) and Sequential Pairing Algorithm (SPA). Results show that the error rate for SPA RO pairs i.e. 1.4% is much less in comparison with neighboring-ROs based pairs i.e. 7%. The Hamming weight of the PUF response in both cases is $\approx 50\%$. The average inter-Hamming distance of the PUF, evaluated on 9 boards, is $\approx 49\%$. Note that all measurements are taken at room-temperature. Ongoing work includes study of the effects of aging, temperature and supply voltage variations on the PUF response.

Based on our evaluation, the implemented RO PUF generates responses with an error rate i.e., $p_{\text{b}} = 1.4\%$, when SPA is applied. In order to keep an additional safety margin a p_{b} of 2% is assumed. The response bits are assumed to be independent, with full entropy, as each RO is used just once. Table 3 presents three different error correcting codes, namely a Reed Muller (RM), Repetition and BCH code with the given parameters. P_{Fail} is obtained by plugging in the parameters values of the respective error correction code and p_{b} in Equation 1. The number of source bits is obtained by calculating the rounds necessary to encode a 128 bit seed. It can be seen that the selected BCH and repetition codes fulfill the failure rate criteria, i.e., $P_{\text{Fail}} \leq 10^{-6}$. Moreover, $BCH(31, 6, 15)$ is also efficient in terms of the required number of source bits to generate a 128-bit key. The required 662 source bits for the BCH code could be obtained using 1344 ROs and SPA. Therefore, we implemented a HDA based on the $BCH(31, 6, 15)$ code in a code-offset construction. The resulting 128-bit key can be used for the implementation of sensed data attestation, and secure boot of the sensor controller modules.

6.2 Preliminary Evaluation

Table 4 provides the FPGA logic area utilization of the 128-bit RO PUF-based key generation module on a Zynq7010 SoC. For reference, the FPGA area utilized by the image readout of the VSN (see Figure 4) is also given. Consid-

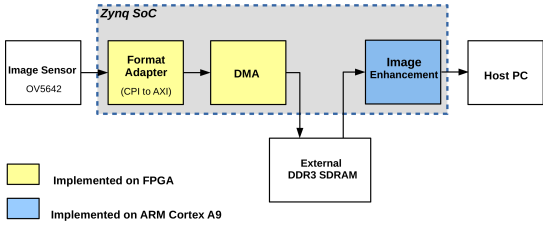

Figure 4: High level architecture of ProSecCo visual sensor node.

	Registers	Look Up Tables
Available	35,200	17,600
PUF Utilization	11%	22%
VSN Utilization	24.2%	40.7%

Table 4: Utilization of the implemented RO-PUF solution on the FPGA.

ering that the RO PUF is the most expensive PUF source in terms of logic area consumption, this initial comparison indicates that PUF based key generation consumes an insignificant amount of the total resources of a typical visual sensor node. The choice of the SRAM PUF, the sensor PUF and other PUF sources mentioned in Section 4.1.1 can save this logic area on the sensor node. However, after complete implementation and integration of sensed data attestation, and secure boot of sensor controller modules, a detailed comparison of resource consumption is possible. Our goal is to show that PUF-based trusted functionality can be achieved in resource constrained sensors nodes without any hardware modifications and without consuming significant amount resources of the sensor node.

7. CONCLUSION

In this work, we presented a trusted sensor design which leverages PUF as root of trust and offers trusted functionality. We term a sensor reading trusted if it is accompanied by integrity, authenticity, and non-repudiation guarantees on the sensed data, associated location and time-stamp. We addressed the problem of potential manipulation of mobile sensors' readings by exploiting vulnerabilities of mobile device OS in participatory sensing applications by proposing incorporation of the trusted sensors for mobile devices. Besides participatory sensing, there are plenty of scenarios where deployment of trusted sensors can enhance the reliability of offered services. Examples include safety related communication in vehicular networks, identification of electricity theft in power distribution networks, surveillance, reliable citizen journalism, and biometric authentication for mobile payment applications. Initial results from our trusted VSN prototype implementation shows that the proposed solution for trustworthiness consumes insignificant resources when compared with total resources of a typical VSN.

Acknowledgment

This research has been funded by the Austrian Research Promotion Agency (FFG) under grant number 842432. Michael

Höberl was supported by the FP7 research project MATTHEW under grant number 610436.

8. REFERENCES

[1] Enhanced Complete Ambient Assisted Living Experiment, 2015. [Online; accessed 11-March-2016].

[2] Search engine for multimedia environment generated content (SMART), 2015. [Online; accessed 11-March-2016].

[3] WikiCity Rome: Senseable City Lab MIT, 2015. [Online; accessed 11-March-2016].

[4] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh. Mobile device identification via sensor fingerprinting. *arXiv preprint arXiv:1408.1416*, 2014.

[5] Y. Cao, L. Zhang, S. S. Zalivaka, C. Chang, and S. Chen. CMOS image sensor based physical unclonable function for coherent sensor-level authentication. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62(11), 2015.

[6] J. Delvaux and I. Verbauwhede. Key-recovery attacks on various RO PUF constructions via helper data manipulation. In *Proceedings of the Conference on Design, Automation & Test in Europe*. EDAA, 2014.

[7] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1), 2008.

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

[9] E. Fonseca, A. Festag, R. Baldessari, and R. L. Aguiar. Support of anonymity in vanets-putting pseudonymity into practice. In *Proceedings of the IEEE Wireless Communications and Networking Conference*. IEEE, 2007.

[10] R. W. Gardner, S. Garera, and A. D. Rubin. Detecting code alteration by creating a temporary memory bottleneck. *IEEE Transactions on Information Forensics and Security*, 4(4):638–650, 2009.

[11] B. Gassend, D. Clarke, M. V. Dijk, and S. Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*. ACM, 2002.

[12] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls. Physical unclonable functions and public-key crypto for FPGA IP protection. In *Proceedings of the International Conference on Field Programmable Logic and Applications*. IEEE, 2007.

[13] M. Höberl, I. Haider, and B. Rinner. Towards a secure key generation and storage framework on resource-constrained sensor nodes. In *Proceedings of the NextMote Workshop 2016*. ACM, 2016.

[14] D. E. Holcomb, W. P. Burleson, and K. Fu. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *Proceedings of the Conference on RFID Security*, 2007.

[15] M. Jakobsson and K. A. Johansson. Retroactive detection of malware with applications to mobile platforms. In *Proceedings of 5th USENIX Workshop on Hot Topics in Security*. USENIX, 2010.

[16] X. Kovah, C. Kallenberg, C. Weathers, A. Herzog, M. Albin, and J. Butterworth. New results for timing-based attestation. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2012.

[17] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. V. Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *Proceedings of the Symposium on VLSI Circuits*. IEEE, 2004.

[18] R. Maes and I. Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. In *Towards Hardware-Intrinsic Security*. Springer, 2010.

[19] J. Petit, F. Schaub, M. Feiri, and F. Kargl. Pseudonym schemes in vehicular networks: A survey. *IEEE Communications Surveys & Tutorials*, 17(1):228–255, 2015.

[20] M. Potkonjak, S. Meguerdichian, and J. L. Wong. Trusted sensors and remote sensing. In *Proceedings of the 9th Annual IEEE Conference on Sensors*. IEEE, 2010.

[21] K. Rosenfeld, E. Gavas, and R. Karri. Sensor physical unclonable functions. In *Proceedings of the International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2010.

[22] S. Saroiu and A. Wolman. I am a sensor, and I approve this message. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*. ACM, 2010.

[23] A. Schaller, T. Arul, V. v. d. Leest, and S. Katzenbeisser. Lightweight anti-counterfeiting solution for low-end commodity hardware using inherent PUFs. In *Proceedings of the 7th International Conference on Trust and Trustworthy Computing*. Springer, 2014.

[24] B. Security. Android FakeID Vulnerability, 2015. [Online; accessed 11-March-2016].

[25] B. Security. Android Master-Key Vulnerability, 2015. [Online; accessed 11-March-2016].

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

[27] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007.

[28] A. Van Herrewege, A. Schaller, S. Katzenbeisser, and I. Verbauwhede. DEMO: Inherent PUFs and secure PRNGs on commercial off-the-shelf microcontrollers. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications security*. ACM, 2013.

[29] O. Vermesan and P. Friess. *Internet of things: converging technologies for smart environments and integrated ecosystems*. River Publishers, 2013.

[30] T. Winkler and B. Rinner. Securing embedded smart cameras with trusted computing. *EURASIP Journal on Wireless Communications and Networking*, 2011.

[31] C. Yin and G. Qu. LISA: Maximizing RO PUF's secret extraction. In *Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2010.