Securing Cloud-Based IoT Applications with Trustworthy Sensing

Ihtesham Haider(✉) and Bernhard Rinner

Alpen-Adria-Universität, Klagenfurt, Austria
{ihtesham.haider,bernhard.rinner}@aau.at

Abstract. The omnipresence of resource-constrained sensors connected to the cloud has enabled numerous Internet of Things (IoT) applications. However, the trust in these IoT applications is severely compromised by security concerns. We introduce a lightweight and effective security approach for such applications by protecting the sensors. Our approach leverages Physically Unclonable Functions (PUF) on the sensor platform to ensure non-repudiation of sensed data and integrity of sensor hardware and firmware. We compare the performance of different PUF implementations on Atmel, ARM, and FPGA-based sensing platforms, analyze the security properties of the proposed protocols and determine the overhead in terms of latency, storage and logic area. Our evaluation shows that it only incurs insignificant overhead on low-end sensors.

1 Introduction

Ubiquitous sensor networks together with cloud computing and storage have played a vital role in enabling numerous IoT applications permeating in domains such as health-care, environmental monitoring, natural disaster detection, and urban planning. A generic IoT application infrastructure comprises spatially distributed sensor nodes, a cloud-based server that collects the sensed data from the nodes, processes the collected data to extract contextual information that is offered as the service to the end-users. The growing software stack on today's sensor nodes and widespread use of public networks (e.g., Internet) for communications make these cloud-based applications more vulnerable and more attractive for attackers. Thus, the reliability of the services offered by these applications critically depends on the "trustworthiness" of the sensed data.

A typical sensor node comprises a sensing unit (also referred to as a sensor) that is connected to a host processor. A sensing unit typically consists of sensing circuitry and a lightweight MCU (known as the sensor controller), whereas a host processor is mostly a powerful processor that runs an operating system (OS). On top of the OS, the specific applications are deployed. *Sensed data pollution* attacks [1–3], that aim to manipulate and fabricate sensors' readings, can be launched using either software or hardware of the sensor node as illustrated in Fig. 1. An adversary may exploit security bugs (e.g., Android Fake ID and MasterKey vulnerabilities) to inject malware in the host OS, physically tamper with the sensor hardware or falsify sensor data by modifications of the sensor

A. Longo et al. (Eds.): IISSC 2017/CN4IoT 2017, LNICST 189, pp. 218–227, 2018.
https://doi.org/10.1007/978-3-319-67636-4_23

firmware [3]. Preventing these data pollution attacks is a major challenge in cloud-based sensing applications.

In a traditional holistic security approach, the entire sensor node starting from the applications down to the operating system and the hardware must be secured. A major limitation of this approach is that security is tightly interwoven with the application logic and the increasing complexity of the system components such as OS, network stack, and system libraries which makes it difficult to achieve tight security guarantees about the node. This work follows an alternative approach that makes security protection inherent property of the sensing unit, which we introduced earlier in our concept paper [4]. The "trustworthiness" of sensed data is improved by ensuring non-repudiation of sensed data and integrity of the sensor hardware and firmware as suggested in [3].

Our lightweight and effective approach leverages sensor-based Physically Unclonable Functions (PUF) and the sensor controller for securing the sensors and the sensed data against the sensed data pollution attacks for cloud-based IoT applications. The contributions of this work are twofold: First, we exploit sensor-based PUFs in combination with lightweight security mechanisms to ensure non-repudiation on sensed readings and integrity of sensor's hardware and firmware. Second, we evaluate our scheme for different sensors and compare the performance of different PUF implementations on Atmel, ARM and FPGA-platforms. The marginal overhead in terms of storage, latency and logic area makes our scheme applicable for various resource-limited sensing platforms.

We assume that an adversary (i) can inject malware in the host device OS that can manipulate or fabricate the sensed data, (ii) has access to the sensors to mount any physical (invasive or non-invasive) attacks and (iii) can modify the sensor firmware statically. We argue that there is not enough economic motivation for the attacker to mount sophisticated attacks such as runtime firmware modification on each sensor. The goal here is to ensure that (i) an untrusted OS running on the host device cannot interfere with the sensors' readings and (ii) compromised sensors are unable to contribute sensed data without being detected.

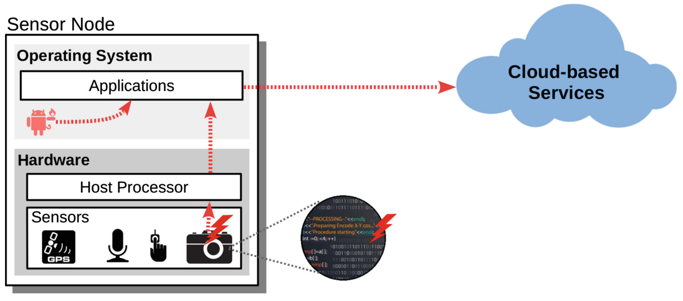

Fig. 1. A high-level sensor node stack, depicting sensed data pollution attacks due to vulnerable host OS, hardware tampering and firmware modification of the sensors.

The rest of the paper is organized as follows: Sect. 2 reviews the state of the art in trustworthy sensing. Section 3 introduces the proposed scheme and provides detailed constructions of the building blocks to achieve non-repudiation of sensed data and integrity of sensor hardware and firmware. Section 4 evaluates our scheme in terms of required logic area, latency and storage. Section 5 concludes this paper.

2 Related Work

Research on trustworthy sensing has mainly focused on the integration of trusted platform modules (TPM) and other secure cryptoprocessors to the sensors or host devices. The anonymous attestation feature of TPM is used to attest the sensed data. Early work on trustworthy sensing [2] was motivated by participatory sensing. TPM was proposed for mobile devices to attest the sensed data. Saroiu and Wolman [1] 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. [5] used TPM to secure embedded camera nodes. Potkonjak et al. [6] proposed an alternative approach for the trusted flow of information in remote sensing scenarios that employed public physically unclonable functions (PPUFs). Despite similar names, PUFs and PPUFs are fundamentally different primitives. PPUFs are hardware security primitives which can be modeled by algorithms of high complexity where as PUFs cannot. The security of PPUF relies on the fact that the PPUF hardware output is many orders faster than its software counterpart. The main drawback of this approach is that current PPUF designs involve complex circuits requiring high measurement accuracy which slows down the authentication process and therefore it is not a scalable solution. Interestingly, some recent research efforts have lead to successful identification of PUF behavior on some sensors. Rosenfeld et al. [7] introduced the idea of a sensor PUF, whereby the PUF response depends on the applied challenge as well as the sensor reading.

The incorporation of TPMs into sensors and host devices requires extensive hardware modifications and introduces significant overhead. Despite the widespread deployment of TPMs in laptops, desktops, and servers for over a decade, TPMs have not yet found their way into sensors or embedded host devices. Protocols based on complex PPUF primitives are slow, and have limited scalability.

3 PUF-Based Trusted Sensors

This section presents our security scheme, which aims for two security objectives: (i) non-repudiation of sensed data and (ii) integrity of sensor firmware and hardware. Non-repudiation of sensed data is ensure by *PUF-based Cert-IBS* module and is implemented in sensor firmware where as the *Verified Boot* ensures the integrity of this firmware. A key element of any security solution is secure key storage. Our scheme leverages a lightweight PUF framework to bind a unique

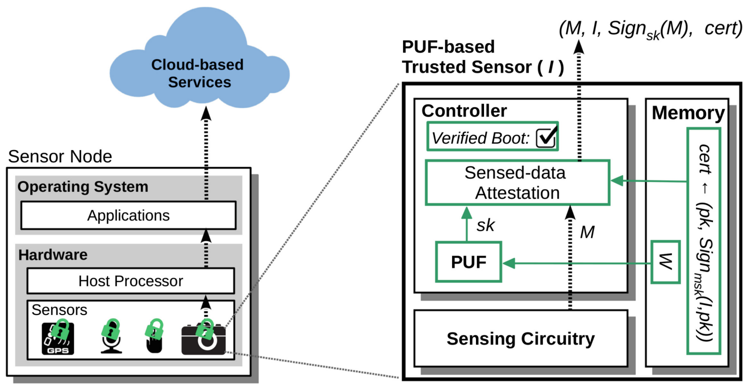

Fig. 2. Our scheme for trustworthy sensing. The readings are signed inside the sensor using a key that is inseperably bound to the sensor by the on chip PUF. The sensor-based security modules are marked in green. (Color figure online)

key to each sensor. All the three components of our scheme: *PUF-based Secure Key Generation & Storage Framework*, *PUF-based Cert-IBS* and *Verified Boot* are realized on the sensor as depicted in Fig. 2 and we refer to the resulting sensor as *PUF-based Trusted Sensor*. Each component requires an enrollment phase, where an interactive protocol is performed between a trusted authority and the sensor before the sensor is deployed in the field.

3.1 PUF-Based Key Generation and Storage Framework

PUFs are lightweight hardware security primitives which typically exhibit a challenge-response behavior. When queried with a challenge c, the PUF generates a response r, that depends on c and the uncontrollable CMOS manufacturing variations of the underlying hardware. Randomness, uniqueness, physical unclonability and tamper resistance properties [8] make PUFs interesting candidates for secure key generation and storage. PUF response is noisy due to variations in the environmental and operating conditions. This noise is measured in terms of intra-Hamming distance (HD^{intra}). The randomness of a PUF response is measure by the Hamming weight (HW). For key generation and storage, PUF response should have zero noise ($\text{HD}^{\text{intra}} \approx 0\%$) and follow uniform random distribution ($\text{HW} \approx 50\%$). Additionally our *PUF-based Cert-IBS* requires the ability to bind an external key to the sensor hardware using PUF. To meet these requirements, we use a PUF-based framework proposed by Tuyls et al. [9] that is able to securely store an external key by masking it with a PUF response. We assume that the PUF is instantiated on the sensor by a legitimate trusted authority. Furthermore, the communication between the PUF and the sensor is secure and is not accessible to the attacker. The framework consists of two phases: enrollment and reconstruction, depicted in Table 1.

Enrollment. This phase is carried out only once by a legitimate authority in a trusted environment to generate helper data W. A challenge c is applied to the PUF and response r is obtained. A random key $k \in \{0,1\}^k$ is chosen, and helper-data is calculated as $W \leftarrow r \oplus C_\mathrm{k}$, where C_k is a code-word chosen from the error-correcting code $\mathcal{C}$, with $2^k - 1$ code-words. W is integrity protected public information.

Reconstruction. It is performed every time the PUF-based key is required. The PUF is subjected to the same challenge c and noisy response r' is obtained. If r' corresponds to the same challenge c applied to the same PUF, k is obtained after decoding using W, otherwise an invalid code-word is obtained. Note that to generate the key k, the sensor has to perform only an XOR and a decoding operation.

Table 1. Framework to bind an externally generated key with a PUF.

Enrollment: $W \leftarrow Gen(r,k)$	**Reconstruction:** $k \leftarrow Rep(r',W)$
Pick a random key: k	$C_{\mathrm{k}'} \leftarrow r' \oplus W$
$C_\mathrm{k} \leftarrow \mathtt{Encoding}(k)$, where	$k \leftarrow \mathtt{Decoding}(C_{\mathrm{k}'})$, if
$C_\mathrm{k} \in \mathcal{C}$, the error-correction code	$\mathtt{Hamming\ distance}(C_\mathrm{k}, C_{\mathrm{k}'}) \leq t$
$W \leftarrow r \oplus C_\mathrm{k}$	t is error-correction capacity of $\mathcal{C}$

3.2 PUF-Based Cert-IBS

Our PUF-based Cert-IBS is based on the definition [10] for constructing a certificate-based IBS scheme using a standard signature scheme (SS). Moreover, it uses the PUF-framework of Sect. 3.1 for secure key storage. We assign each PUF-enabled sensor an identity I which can be any physical identifier such as the sensor's serial number or EPC. The PUF-based Cert-IBS works in two phases: *enrollment*, performed once by the trusted authority before deployment and *sensed data attestation*, performed by the sensor every time it senses fresh data. PUF-based Cert-IBS is a tuple (MK, UK, $SIGN$, VER) of polynomial time algorithms where MK, UK, $SIGN$ and VER refer to master key generation, user key generation, signing and verification algorithms respectively. Let the standard signature scheme $SS := (K, Sign, Ver)$ where K, $Sign$ and Ver are the key generation, signing and verification algorithms. Then the PUF-based Cert-IBS (MK, UK, $SIGN$, VER) is associated to a standard signature scheme SS (K, $Sign$, Ver) as follows:

Enrollment. The trusted authority runs the K of SS as the MK to generate the master key pair: $(mpk, msk) \leftarrow MK(1^k)$. During enrollment, the authority binds a unique key with the PUF-enabled sensor using the user key generation algorithm UK as follows: The sensor presents itself to the authority using identity I. The authority requests the sensor to subject its PUF with challenge c. The sensor obtains the corresponding PUF response r and provides it to the authority who then (i) runs the K of SS and generates a key pair (pk, sk) for the sensor (ii) determines the helper data W by executing $Gen(r, sk)$ of Table 1 and (iii) creates a certificate on the identity and public key of the sensor using signing algorithm $Sign$ of the SS i.e., $cert \leftarrow (pk, Sign_{msk}(pk, I))$. Helper data W and the certificate $cert$ are stored in the sensor memory. The user key usk is given by the PUF-bound secret key sk and the certificate $cert$ (see Fig. 3 (left)).

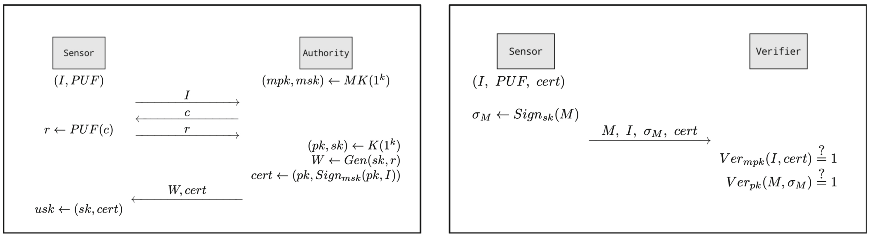
Sensor
Authority
(I, PUF)
$(mpk, msk) \leftarrow MK(1^k)$
I
c
$r \leftarrow PUF(c)$
r
$(pk, sk) \leftarrow K(1^k)$
$W \leftarrow Gen(sk, r)$
$cert \leftarrow (pk, Sign_{msk}(pk, I))$
$W, cert$
$usk \leftarrow (sk, cert)$
Sensor
Verifier
$(I,\ PUF,\ cert)$
$\sigma_M \leftarrow Sign_{sk}(M)$
$M,\ I,\ \sigma_M,\ cert$
$Ver_{mpk}(I, cert) \overset{?}{=} 1$
$Ver_{pk}(M, \sigma_M) \overset{?}{=} 1$

Fig. 3. Enrollment (left) and sensed data attestation (right) phases of PUF-based Cert-IBS scheme

Sensed Data Attestation. Sensed data attestation is performed every time before the sensor outputs a new reading. The private key required for signing is reconstructed at the power-up as $sk \leftarrow Rep(r', W)$ (see Table 1). To sign a new reading M, the sensor uses $Sign$ of SS to obtain $\sigma_M \leftarrow Sign_{sk}(M)$. The PUF-based Cert-IBS signature on the sensed data is given by $SIGN(M) := (I, \sigma_M, cert)$. The verification algorithm VER returns 1 (SUCCESS) if $Ver_{pk}(M, \sigma_M) = Ver_{mpk}(I, cert) = 1$, where Ver is the verification algorithm of SS (see Fig. 3 (right)). If verification of either the certificate or the sensed data fails, the reading is considered corrupt and the protocol is aborted.

The PUF-based Cert-IBS scheme enables a sensor to ensure integrity and authenticity on each reading, verifiable by the cloud-based application server. Any manipulation or fabrication of sensed data will result in failure during PUF-based Cert-IBS verification. The security of the PUF-based Cert-IBS scheme relies on security of PUFs as secure key storage and security of the underlying SS scheme. Using PUFs, the key is derived from device properties upon device start-up. During the state when device is off, the key exists in form of unreadable CMOS manufacturing variations. The side-channel attacks can be thwarted by breaking the correlation between side-channel information and the secret structure of the PUF. The security of PUF-based Cert-IBS scheme is related to the security of underlying SS scheme via Theorem 1. We omit the proof of Theorem 1 since it is similar to the proof of Theorem 3.5 of [10].

Theorem 1. *Let SS be a uf-cma secure standard signature scheme. Let* PUF-based Cert-IBS *be the corresponding IBS scheme as per construction of Sect. 3.2. Then* PUF-based Cert-IBS *is a uf-cma secure IBS scheme.*

3.3 Verified Boot

In our scheme, the sensed data attestation is performed in the firmware of the sensor controller (see Fig. 2). Therefore, during the *enrollment* phase of the Verified Boot, the trusted authority binds the legitimate firmware (also responsible for the sensed data attestation) with the sensor I using the on-chip PUF as follows: Given a sensor controller with a two stage boot-chain, i.e., the boot-loader and the firmware, the bootloader is modified to additionally compute the message authentication code (MAC) of the legitimate firmware (h_{FW}) and stores it in the immutable memory available on the sensor controller. The scheme assumes that one-time programmable memory such as OTPROM, MROM or PROM is available on the sensor controller. After the sensor is deployed for sensing, Verified Boot verifies the integrity of the sensor firmware at every power-up as follows: the bootloader generates the key, calculates a fresh hash value of the firmware and compares it with the reference hash value stored in the ROM during enrollment. Verified Boot resists against sensor firmware modification attacks. If a modification in the sensor firmware is detected during power-up, the boot process is aborted.

3.4 Security Properties

The correctness of the trustworthy sensing scheme follows from the correctness of the PUF-based Cert-IBS scheme. Since, any compromise to the trusted authority nullifies the trust and non repudiation guarantees on the sensed data, we emphasize the *offline* nature of the authority in our scheme greatly reduces the risk of compromise. The scheme withstands the sensed data corruption attacks due to (i) compromised host device OS (ii) tampered sensor hardware and (iii) modified sensor firmware. The host OS receives signed sensor readings and the corresponding certificate. In order to inject fabricated data at OS level, an attacker has to produce a valid signature certificate pair i.e., valid PUF-based Cert-IBS signature. A uf-cma secure PUF-based Cert-IBS implies that there is negligible probability that an attacker produces a valid PUF-based Cert-IBS signature. Since the PUF behavior corresponds to underlying hardware, given a tampered sensor hardware, PUF-based key generation implies that this results in generation of invalid secret key leading to generation of invalid signatures. Lastly, the offline attacks to modify the sensor firmware are detected by the Verified Boot.

4 Implementation and Evaluation

This section discusses the overhead incurred by our approach on a sensor in two parts. In Sect. 4.1, we present the implementation results of our PUF-based Key Generation & Storage Framework. Section 4.2 evaluates the total overhead on the sensor incurred by PUF-based Key Generation & Storage Framework, PUF-based Cert-IBS and Verified Boot modules in terms of storage, logic-area and latency.

4.1 PUF-Based Key Generation and Storage Framework

Various PUF sources are inherent to a typical sensor including SRAM PUF, ring oscillator (RO) PUF, and sensor-specific PUFs [7,11,12]. We aim to identify PUF sources that are commonly available on most of the sensors e.g., SRAM PUF and RO PUFs. We implemented PUFs on three platforms of varying complexities: (i) Atmel ATMEGA328P, a lightweight 8-bit MCU, (ii) ARM Cortex M4, a 32-bit MCU, and (iii) Xilinx Zynq7010 SoC with re-programmable logic and a dual core ARM Cortex A9. These platforms are perfectly suitable as sensor controllers for a wide range of sensors. The power-up state of the SRAM cells on the ATMEGA328P and the ARM Cortex M4 show PUF behavior. Figures 4(a) and (b) depicts the error-rate (measured as intra-Hamming distance (HD^{intra})) and the non-uniform distribution (measured as Hamming weight (HW)) of 100 PUF-responses obtained at room temperature. We implemented the RO PUF in FPGA area of Xilinx's Zynq7010 SoC. We characterized the RO PUF for HD^{intra} and HW, from 800 responses obtained over a temperature range of 0–60°C, depicted in the Fig. 4(c). The maximum error-rate HD^{intra}(max) for the three PUFs $\approx$ 7.2%, 9.16%, and 6.97%. So, we designed the framework of Table 1 that can correct 10% error-rate. The error-correcting code determines the number of required PUF response bits and hence the size of PUF, so we experimented with two code: (i) a simple code: BCH(492, 57, 171) and (ii) a concatenated code: Reed Muller(16, 5, 8) $\|$ Repetition(5, 1, 5). The resources consumed by SRAM PUF-based framework on ATMEGA328P MCU (Arduino board) and RO PUF-based framework on Xilinx's Zynq7010 SoC (MicroZed board) for 128-bit key are summarized in Table 2.

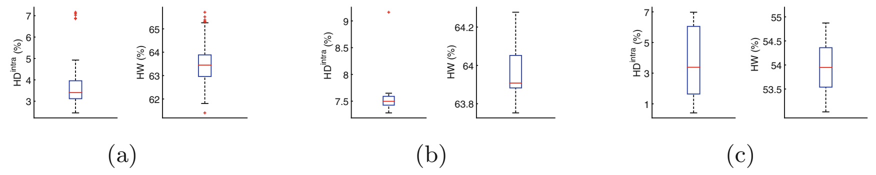

Fig. 4. PUF characterization of (a) 1 kB SRAM on Atmel ATMEGA328P 8-bit MCU, (b) 15 kB SRAM on ARM Cortex M4 32-bit MCU, and (c) RO PUF comprised of 1040 3-stage ring oscillators implemented in FPGA part of Zynq7010 SoC. The PUF quality parameters for (a) and (b) are calculated over 100 PUF responses at the room temperature and for (c) 800 responses taken over temperature range 0–60°C. The mean and max. error-rate (HD^{intra}) for (a) $\approx$3.4% and 7.2%, (b) $\approx$7.66% and 9.16%, and (c) $\approx$3.6% and 6.97%. The randomness of PUFs (HW(mean)), for (a) $\approx$63.5%, (b) $\approx$63.96%, and (c) $\approx$53.95%.

Table 2. Implementation results of the PUF-based 128-bit key generation and storage framework for the sensors

PUF source	Error correcting code	Area (≈ logic gates)	Latency (Key reconstruction)	Storage Helper data (W)
RO	BCH	2210	≤100 ms	**1105** bits
	RM || Rep	3456		1728 bits
SRAM	BCH	NA	≈30 ms	**1105** bits
	RM || Rep	NA		2048 bits

4.2 Sensor Overhead

Our prototype PUF-based trusted sensor of Fig. 2 comprises OV5642 image sensor and Zynq7010 SoC as the sensor controller. We evaluate the storage, logic-area and latency overhead and summarize the results in Table 3.

Table 3. Logic-area, latency, and storage requirements of our scheme on a sensor

Area RO PUF only	Latency Sensed data attestation	Storage $W + cert + h_{FW}$
2210 logic gates	6.27 ms	1105 + 480 + 256 bits (≈230 bytes)

Storage Requirements. The sensor I needs to store (i) helper data W and certificate *cert* of PUF-based Cert-IBS and (ii) firmware hash value h_{FW} for the Verified Boot. For RO PUF-based framework, helper data $W \approx 1105$ bits. The *cert* is comprised of *pk* and $Sign_{msk}(I, pk)$. We implemented BLS signature scheme where $pk \approx 320$ bits and $Sign_{msk}(I, pk) \approx 160$ bits which amounts to 480 bits. h_{FW} is a 256-bit hash value computed using SHA-256. Therefore, the total storage requirement on a sensor for asymmetric version of our scheme is not more than 1841 bits (≈230 bytes).

Latency. PUF-based Key Generation and Verified Boot are performed at start-up and therefore run-time delay overhead is only incurred by the sensed data attestation phase of PUF-based Cert-IBS scheme. For sensed data attestation, we used the open-source *pairing-based cryptography library* and measured the latency of 6.27 ms on ARM cortex A9 core of Zynq7010 SoC. At 640×480 resolution, OV5642 can provide up to 15 FPS, which implies that a new frame is available for signing every 66.7 ms $\gg$ 6.27 ms of the signing latency.

Logic-Area Overhead. The logic-area is required only if the RO PUF is implemented. From Table 2, 2210 logic gates are used to generate a 128-bit key.

5 Conclusion

Security concerns in cloud-based IoT applications present severe obstacles for widespread adoption of these applications. In this paper, we presented a PUF-based scheme to secure the sensed data at its source, in order to improve the trust in the services provided by these application. The scheme is proven lightweight resulting in very low overhead wrt. storage ($230B$), and latency (6.27 ms). The logic area overhead can be avoided by choosing a PUF source inherent of the sensor (e.g., SRAM PUF or sensor PUF). This is significant improvement over TPM-based approach [2] that incurs an overhead of a secure co-processor chip on each sensor and takes 1.92 s for the attestation.

Acknowledgment. This research has been funded by the Austrian Research Promotion Agency (FFG) under grant number 842432. Michael Höberl has implemented RO PUF and was supported by the FP7 research project MATTHEW under grant number 610436.

References

1. Saroiu, S., Wolman, A.: I am a sensor, and i approve this message. In: Proceedings of Mobile Computing Systems & Applications, pp. 37–42. ACM (2010)
2. Dua, A., Bulusu, N., Feng, W.-C., Hu, W.: Towards trustworthy participatory sensing. In: Proceedings on Hot topics in security, p. 8. USENIX (2009)
3. Kapadia, A., Kotz, D., Triandopoulos, N.: Opportunistic sensing: security challenges for the new paradigm. In: Proceedings of Communication Systems and Networks and Workshops, pp. 1–10. IEEE (2009)
4. Haider, I., Höberl, M., Rinner, B.: Trusted sensors for participatory sensing and iot applications based on physically unclonable functions. In: Proceedings of Workshop on IoT Privacy, Trust, and Security, pp. 14–21. ACM (2016)
5. Winkler, T., Rinner, B.: Securing embedded smart cameras with trusted computing. EURASIP J. Wirel. Commun. Netw. **2011**, 530354 (2011)
6. Potkonjak, M., Meguerdichian, S., Wong, J.L.: Trusted sensors and remote sensing. In: Proceedings on Sensors. IEEE (2010)
7. Rosenfeld, K., Gavas, E., Karri, R.: Sensor physical unclonable functions. In: Proceedings on Hardware-Oriented Security and Trust (HOST). IEEE (2010)
8. Maes, R.: Physically unclonable functions: constructions, properties and applications. Ph.D. dissertation, University of KU Leuven (2012)
9. Tuyls, P., Batina, L.: RFID-tags for anti-counterfeiting. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 115–131. Springer, Heidelberg (2006). doi:10.1007/11605805_8
10. Bellare, M., Namprempre, C., Neven, G.: Security proofs for identity-based identification and signature schemes. J. Cryptol. **22**(1), 1–61 (2009)
11. Cao, Y., Zhang, L., Zalivaka, S.S., Chang, C., Chen, S.: CMOS image sensor based physical unclonable function for coherent sensor-level authentication. IEEE Trans. Circu. Syst. I Regular Papers **62**(11), 2629–2640 (2015)
12. Rajendran, J., Tang, J., Karri, R.: Securing pressure measurements using SensorPUFs. In: Proceedings of Circuits and Systems, pp. 1330–1333. IEEE (2016)