How to Manage Keys and Reconfiguration in WSNs Exploiting SRAM Based PUFs

Domenico Amelino, Mario Barbareschi, Ermanno Battista and Antonino Mazzeo

Abstract A wide spectrum of security challenges were arose by Wireless Sensor Network (WSN) architectures and common security techniques used in traditional networks are impractical. In particular, being the sensor nodes often deployed in unattended areas, physical attacks are possible and have to be taken into account during the architecture design. Whenever an attacker enters in possession of a node, he/she can jeopardize the network by extracting cryptographic keys used for secure communication. Moreover, an attacker can also try to brute force the keys, hence they should be fully random and hard to guess. In this paper, we propose a novel solution based on generating keys from unique physical characteristics of a node integrated circuit without requiring additional hardware compared to common WSN node architectures. To this aim, we exploit the Static Random Access Memory based Physically Unclonable Functions and we show their applicability to the WSN by implementing a working prototype based on the STM32F4 microcontroller.

Keywords Wireless Sensor Network · Physically Unclonable Function · Static Random Access Memory · Reconfiguration

D. Amelino · M. Barbareschi (✉) · E. Battista · A. Mazzeo
DIETI - Department of Electrical Engineering and Information Technologies, University of Naples Federico II, Via Claudio 21, 80125 Naples, Italy
e-mail: mario.barbareschi@unina.it

D. Amelino
e-mail: domenico.amelino@unina.it

E. Battista
e-mail: ermanno.battista@unina.it

A. Mazzeo
e-mail: antonino.mazzeo@unina.it

M. Barbareschi · E. Battista · A. Mazzeo
CeRICT scrl - Centro Regionale Information Communication Technology, Naples, Italy

G. De Pietro et al. (eds.), *Intelligent Interactive Multimedia Systems and Services 2016*, Smart Innovation, Systems and Technologies 55,
DOI 10.1007/978-3-319-39345-2_10

1 Introduction

Wireless Sensor Networks (WSNs) are commonly devoted to collect environmental data which need to be transferred to a collection/central node for further processing and analysis. Security has a central role for WSN applications, data have to be transferred to central nodes is a secure way so as not be altered or crafted by malicious attackers. Due to the inherent wireless nature of WSN nodes, the computational resources available are scarce and security mechanisms must introduce a negligible overhead [2, 3, 7]. Integrity, confidentiality and authentication are commonly offered by means of symmetric cryptography where the data payload is encrypted/decrypted only by pairs of sensing node—collector. In such context, multiple keys are usually enforced so as to create a dedicated channel between a sensing node and the collector: each node has a key for encrypting/decrypting messages to/from the central node. The central node has to know the symmetric key for each sensing node.

This solution is actually not so secure if the node is subject to physical capture by malicious actors. Whenever the node is captured, the key can be extracted from its memory, and this allows the attacker to impersonate a legitimate node because he/she can gather all the information and security mechanisms stored in the node. To deal with this issue, the technology used for the key storage must be hardened to make worthless any access attempt. Moreover, involved keys have to characterized by a great randomness and hard to guess.

A novel approach is based on the generation of security keys from device intrinsic physical characteristics (like humans fingerprint) rather than storing this information in a non-volatile memory (NVM). The great advantage of intrinsic physical characteristics is their strict coupling with the device. Basing on this concept the literature defined the silicon Physically Unclonable Functions (PUFs) primitives, which compute a string of bits based on measurements of a physical parameter of the integrated circuit (IC). This string is know as *response* and it is intrinsically random and unique so as to be used to identify a device.

For these reasons, in this paper, we propose a key *generation* mechanism based on exploiting the PUF security properties. In particular, we describe the kind of PUF that are suitable for the WSN domain, namely the Static Random Access Memory (SRAM) PUF, we detail the process to generate a secret key from PUF responses and then we provide a working prototype deployed on a commercial microcontroller widely used in WSNs. Moreover, we discuss two case study scenarios related to the secure remote reconfiguration of WSN nodes and the symmetric key renewal process. Both of them are based on our PUF architecture capable of offering a secure, anti-tamper and trustworthy perimeter for the user application execution without requiring additional hardware resources.

2 Related Work

PUFs have been recently adopted for the physical protection of embedded devices, since they are able to guarantee attractive security properties in many application domains. As for WSNs, research efforts have been focusing on the development of novel authentication mechanisms. A mechanism to prevent nodes cloning is presented in [23]. Authors devise a mutual authentication scheme which relies on the PUF challenge/response mechanism and they demonstrate the resiliency of the approach against clone attack, replay attack, eavesdropping and tampering attempts. Similarly, in [15] a mutual authentication protocol is tailored for Wireless Body Sensor Networks, by adopting PUFs and one-way hashing functions. Those solutions, although presenting actual schemes based on PUFs and intended for WSNs, are not actualized by a concrete realization of a real PUF architecture.

Conversely, Liu et al. in [16] exploit PUFs for a trustworthy key generation. The technology which the PUF circuit is based on is the SDRAM type 3 (DDR3): indeed, they use the decay signature of the memory cells with one transistor/one capacitor structure. Commonly, wireless sensor nodes are resource constrained devices, and for the energy efficiency they do not employ power-consuming memory technologies, such as the DDR one.

On the contrary, this paper introduces the adoption of an SRAM-based PUF for WSNs taking into account that most of micro-controllers used for embedded nodes are inherently equipped with an SRAM. This allows for a security improvement requiring neither hardware substitution nor additional hardware resources, thus making it widely applicable. Our approach offers the same security guarantees of previously presented works and further improves their applicability by showing real use scenarios for node remote reconfiguration and cryptographic key renewal.

The remote reconfiguration is part of the Moving Target Defense (MTD) security approach [1] and we advance it by ensuring trustworthiness on the execution environment of the node. Modern reconfiguration schemes for WSNs, such as SIREN [8], tend to pre-load several application images into an external NVM. Due to the unattended nature of WSNs, nodes are subject to physical capture and, hence, cloning. Therefore, solely employing reconfiguration mechanisms is not completely effective.

It is worth to notice that some attempts to ensure a trustworthy software execution in WSNs have already been presented in [7, 14]. Though, those solutions require special hardware, such as the TPM-enabled hardware, in order to provide a root-of-trust and boot the node through a secure procedure. This is obviously costly due to the technological requirements (for instance, the chip Fritz or tamper resistant hardware) [11], and thus this limits their employment. With out approach we offer an analogous secure boot procedure without additional hardware cost.

3 Background

The opportunity of extracting physical characteristics from fabric induced variability for integrated circuits (ICs) has been representing a momentous breakthrough for the security of electronic devices. Silicon PUFs are circuits able to extract such physically imprinted characteristics, producing binary strings, namely *responses* that can be envisioned to be identifiers of ICs pretty much like the human fingerprints or the DNA. Being inherently random, manufacturing variations imprint random and unique physical effects. Moreover, any physical detail or parameter is hidden, i.e. cannot be predicted, and generated by uncontrollable process. Thus, PUFs responses are unique, unclonable and unpredictable. Moreover, PUF circuits are tamper-evident because any tamper attempt will alter the physical characteristics exploited by the PUF dramatically changing the PUF response.

PUFs can be categorized by considering their operational mechanism. Actually, some PUF architectures exploit delay measurements, such as the Arbiter PUF, Ring Oscillator PUF [21] and the Anderson PUF [4]. Other architectures exploit the pattern generated on the start-up of memory cells, such as the SRAM PUF [13] or the STT-MRAM PUF [22].

Interestingly, the SRAM PUF can be potentially exploited as secure primitive for a large amount of devices, as it is a very spread non-volatile memory technology. The mechanism behind the SRAM PUF is the value assumed by each SRAM cell when it is being powered-up (cold start). As shown in Fig. 1, the cell is realized through two symmetric halves, which are two cross-coupled inverters. They generate a feedback which makes the initial voltage point unstable and, hence, forces each cell to move towards one of the two stable voltage values (logic-0 or logic-1). Indeed, the feedback amplifies the differences of voltages between the two symmetrical halves caused by manufacturing variability.

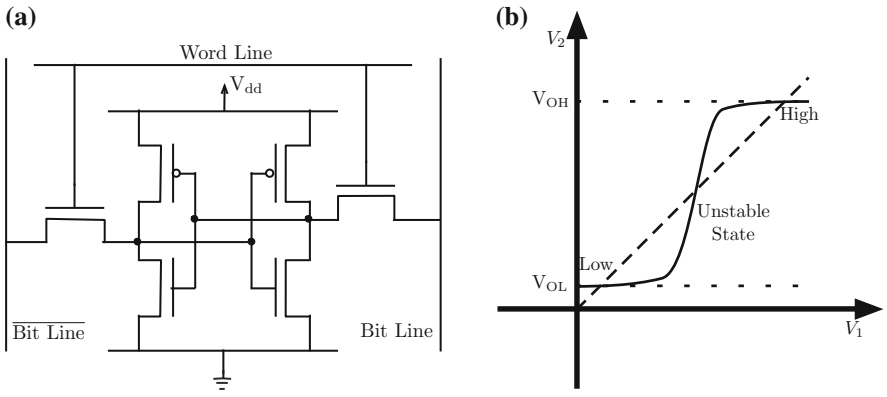

Fig. 1 Details on SRAM cell. **a** Transistor-level schematic of a SRAM cell in CMOS technology. **b** Input-output voltage graph of crosscoupled inverting stages

Ideally, a PUF should always be able to exactly reproduce the same output when queried. However, variations of electrical characteristics caused by environmental conditions, such as temperature variations or power supply instability, might change some response bits [5, 19]. Indeed, as shown in [6], some SRAM cells are less stable than others and more prone to be affected by external conditions.

In order to use the PUF response for cryptographic operations, it is required to have stable responses. The literature introduced post-processing techniques able to produce binary strings, eligible for cryptographic operations, from unstable PUF responses. Majority voting is a post-processing technique based on multiple measurements over a PUF circuit [17]. The technique can be accomplished by averaging N sequential measurements on a PUF (Temporal Majority Voting) or N simultaneous measurements on different PUFs (Spatial Majority Voting). Majority voting requires a threshold to establish the outcome of voting. Whenever the measurements to vote accumulates around the threshold, the outcome is not reliable. Conversely, the fuzzy extractor technique involves only one measurement and exploits an error correction code (ECC) in order to retrieve stable binary strings from noisy responses. In this way, PUF responses, as shown in the next Section, can be used as key provider for cryptographic-schemes.

4 Extracting Keys from Embedded SRAMs

The literature provides a great number of scientific paper, reporting experimental measurements on different SRAM circuits [6, 10]. PUFs guarantee random, unclonable and unpredictable responses these properties fulfill what is required for good cryptographic keys. Though, in order to exploit these benefits, the PUF mechanism has to be enriched with a post-processing technique.

Among recovering techniques previously illustrated, we adopt the fuzzy extractor approach. The fuzzy extractor is a recovery scheme composed of two primitives: (i) information reconciliation, which removes the error generated by the noise, and (ii) privacy amplification, which enhances the information bits entropy (randomness). Both the primitives rely on Helper Data, which is generated during the **enrollment phase**, described in Fig. 2a. A reference PUF response R is extracted and a secret symbol C_s is randomly obtained. Thus, Helper Data W is generated by a *xor* function, while the privacy amplification outputs the key K. To ensure the secrecy of K, this phase must be carried out within a trusted environment, before issuing the device, while Helper Data can be publicly available because it does not disclose secrets exploitable by attackers. Helper Data is used during the reconstruction phase, detailed in Fig. 2b, where K is reconstructed from the response R', which is the PUF response R of the device in the operating environment, thus may be noisy, and hence different. The **reconstruction phase** can be accomplished by the device, which embeds the PUF, at any time by using the Helper Data W.

Moreover, the fuzzy extractor scheme contains an ECC and a cryptographic hash function. The ECC is an algorithm developed to obtain a binary string such that a

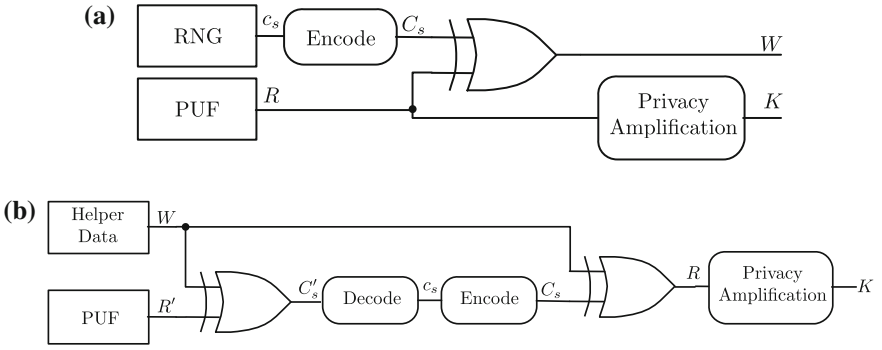

Fig. 2 The fuzzy extractor scheme. **a** Steps of the enrollment phase which generate the key and Helper Data. **b** Steps of the key reconstruction phase which regenerate the same key exploiting Helper Data

certain amount of errors caused by the external environment can be detected and corrected. Figure 2 shows the ECC primitives, namely the encode and decode functions, which are used in both the enrollment and the key reconstruction phases.

The fuzzy extraction design involves the crucial configuration of the ECC scheme. Indeed, an ECC is able to recover a maximum number of errors in the binary string that has to be processed. This amount is obviously correlated with the PUF stability. The maximum number of error t is determined by considering the bit error probability p_b, which specifies the probability that a transmitted information bit is received altered. The noise of PUF responses can be modeled as a binary symmetric channel and the error probability that a block of N bits has more than t error is:

$$P_{block} = 1 - \sum_{i=0}^{t} \binom{N}{i} p_b^i (1 - p_b)^{N-i} \tag{1}$$

Thus, P_{block} represents the failure rate of the decoder correction procedure of an N bits block.

The keys, extracted from PUF responses, have to be hard to guess, hence characterized by a high value of entropy. The entropy is related to the number of independent bits of an information source that can be generated. The entropy can be estimated by calculating the min-entropy, a lower bound of entropy that represents the worst-case measure of uncertainty for a random variable [12].

Let p_{max} be the maximum value between the occurrence probabilities of 0 and 1 bits. The min-entropy can be calculated as $H_{min} = -\log_2(p_{max})$. In particular, as Guajardo et al. assumed in [13], each bit of SRAM is independent, hence the min-entropy for N bits of SRAM can be calculated as sum of min-entropy for each bit:

$$(H_{min})_{total} = \sum_{i=1}^{N} -\log_2(p_{max_i}) \tag{2}$$

The value p_{max_i} can be derived by sampling SRAM bits from different devices. It is possible to calculate the average min-entropy as $\overline{H}_{min} = (H_{min})_{total}/B$, where B is the size of the memory block. The average min-entropy allows to determine the amount of source bits required to derive a significantly random key with a given length. Indeed, for a key of L bits, at least $\lceil L/((\overline{H}_{min})_{total} \rceil$ bits have to be extracted from the PUF.

For instance, to obtain **a key of 128 bits** from a SRAM PUF with an entropy of 0.8, the PUF has to provide **a response of 160 bits**.

5 Case Studies

In order to prove the feasibility for SRAM PUF to be employed in WSNs, this Section details a real implementation of a node based on a commercial microcontroller, namely the STM32F4 device, equipped with the previously illustrated PUF scheme. In particular, we choose the STM32F4 family based on ARM Cortex-M 32-bit microcontrollers, for its widespread availability and suitability for WSN applications. We use the STM32F407VGT6 which includes a ARM Cortex-M4F, 1 MB of NVM flash and 192 KB of SRAM.

First of all, the approach is based on the deployment of a secure bootloader, which represents the root-of-trust for applications that will run on the device. The bootloader has to be stored in the NVM, residing in a secure perimeter: the microcontroller has to be configured to deny any external access to memories. Furthermore, the bootloader is the only handler of the SRAM PUF and the only key manager for the node. This is required to ensure the correct extraction of the PUF response from the pristine state of the SRAM start-up pattern.

The STM32F4 microcontroller offers a memory protection mechanism to secure the perimeter with different memories read/write restrictions [20]. In particular, we adopt the level 2 read protection, which forbids all external accesses to flash sectors and also disables every debug interface, while level 2 write protection prevents flash memory overwrites. A joint use of these two approaches protects the bootloader ensuring its integrity, i.e. trustability.

The role of the bootloader is to load user application images which reside outside the secure perimeter and the PUF mechanism is used to extend trustability to them. Indeed, the bootloader firstly computes the PUF-based key using the fuzzy extraction procedure described in the previous Section. Then, the key is used to decrypt or verify the user application image. In case of success, the bootloader prepares to run the user application: the bootloader cleans the SRAM memory and loads the user software.

In our implementation, we use 128-bits key and a Reed Muller ECC scheme,[1] which requires a PUF response of 2816 bits. The primitives for decrypting and the

[1]The Reed-Muller ECC has a (128,8,63) configuration, which has probability error of 4.321086e-09.

privacy amplification are both implemented with Speck [9], a lightweight block cipher. In particular, in order to provide a hash digest from the Speck encryption primitive, we employed the Davis-Meyer scheme [18].

The bootloader so far described is perfectly suitable for WSNs nodes thanks to its significantly small footprint: the complete bootloader requires just 16 KB, equivalent to the 1.5 % of the total available flash memory of the STM32F407VGT6 device. Moreover, the time overhead introduced by the PUF extraction is about 104 ms.

5.1 Secure Reconfiguration

Being able to reconfigure a node in a WSN is a desirable feature so as to modify over time the specific tasks carried out by a node. Previous works, like in [8], have shown this advantage and the mechanisms to efficiently achieve remote reconfiguration A secure remote reconfiguration can be easily supported by our PUF based mechanism. Traditionally, to reconfigure a WSN node, a set of application images are preloaded once into an external memory at the deployment phase. During the WSN life cycle, as shown by SIREN, it is possible to implement a set of policies to reconfigure a node. The reconfiguration is based on the following phases: (i) choosing the next application to run from the image set; (ii) saving all the information required in the next application in a NVM; (iii) prepare the bootloader with a reference to the chosen next application; (iv) reboot the device. Once the bootloader starts, it will retrieve the pointer to the application to run from NVM and proceed to its loading.

By using the PUF-secured boot process described above, we can ensure that the application images are protected by means of cryptography. In the deployment phase, all the application images have to be encrypted with the device's PUF key extracted in the enrollment phase (Sect. 4) and the loaded in the external NVM that can be outside the secure perimeter. When the device boots, it generates the key to decrypt the application image by applying the reconstruction phase (Sect. 4), decrypts the application image and loads it in RAM, actually starting its execution.

On STM32F407VGT6 microcontroller, the execution time required by the bootloader to extract the key and build the root-of-trust is about *900* ms for an image file of 10 KB.

5.2 Key Renewal

A good security practice is to change the symmetric keys involved in communication over time. To avoid the transmission of keys on non-secure or potentially non-secure channel, a set of keys is pre-loaded on each node of the WSN. By applying policies to trigger the enforcement of another key in the set (e.g. max number of messages per key, time expiration, explicit key change message), they communication key is renewed.

Our PUF based solution, offers a key renewal procedure that generates high-entropy keys from the SRAM to be used by a trusted application running on the system. During device power-up, the boot loader can generate one or more keys by applying reconstruction phase (Sect. 4) on different SRAM areas and storing them into a specific memory location. At each start-up, the key given to the application can be renewed by executing the reconstruction phase on a different SRAM area, that can be configured/selected by using persistent information stored in an external NVM. It follows that the enrollment phase (Sect. 4) had to be accomplished on same SRAM areas in order to provide symmetric keys to the central node (offline procedure executed once at WSN deployments time). It is guaranteed that the software application running on the system is trusted (loaded with a secure boot procedure) and is able to retrieve the generated keys from a memory location conveniently indicated by the bootloader.

The key renewal procedure requires a device restart because it can only be executed on a pristine state of the SRAM start-up pattern. When the application requires to renew the communication key, the device can cold reboot itself.[2] The only penalty added is the actual time spent for reboot that can be considered negligible in most of the WSN sensing applications.

6 Conclusion

The tight constraints and hostile environmental working conditions of WSNs make the conventional computer security techniques inadequate for embedded sensor nodes. The main challenge is related to the unattended nature of the network, which exposes what is installed in the node memories to attacks, including secure protocols and their cryptographic keys. The key management has to guarantee strong keys and a secure perimeter where use them. In this paper, we moved in that direction, providing a key management based on the adoption of an SRAM PUF as key storage and provider. By exploiting the start-up pattern of an SRAM, the fuzzy extractor technique is able to generate bit strings which are eligible to be employed as cryptographic keys. We showed main steps involved in the design of the PUF and two meaningful case studies, focusing on the node reconfiguration and on the key renewal mechanism. The main advantages of our approach are listed below:

- Generated keys are tight coupled with physical parameters of SRAM cells, cannot be cloned and are hard to guess;
- PUF generated keys inherit the randomness from the fabric manufacturing variability and do not require an installation phase as they are retrieved on demand from physical parameters.
- The SRAM Based PUF does not require additional hardware resources and involved software procedures have to be executed once at the start-up.
- The approach requires only an SRAM accessible in a pristine state.

[2]The device goes into standby mode to power down the SRAM, before rebooting.

- The keys are generated by the SRAM PUF on demand and are not stored in an NVM; moreover, they are managed in a secure perimeter.
- The adoption of a PUF opens the possibility to make the execution environment trustworthy.

Acknowledgments The list of Authors is in alphabetical order. The corresponding authors are Mario Barbareschi and Ermanno Battista. This research work was partially supported by CeRICT for the project NEMBO—PONPE_00159.

References

1. Albanese, M., Battista, E., Jajodia, S., Casola, V.: Manipulating the attacker's view of a system's attack surface. In: 2014 IEEE Conference on Communications and Network Security (CNS), pp. 472–480. IEEE (2014)
2. Amato, F., Chianese, A., Moscato, V., Picariello, A., Sperli, G.: Snops: A Smart Environment for Cultural Heritage Applications, pp. 49–56 (2012)
3. Amato, F., Mazzeo, A., Moscato, V., Picariello, A.: Exploiting cloud technologies and context information for recommending touristic paths. Stud. Comput. Intell. **511**, 281–287 (2014)
4. Anderson, J.H.: A puf design for secure fpga-based embedded systems. In: Proceedings of Asia and South Pacific Design Automation Conference, pp. 1–6. IEEE Press (2010)
5. Barbareschi, M., Bagnasco, P., Mazzeo, A.: Supply voltage variation impact on anderson puf quality. In: 2015 10th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), pp. 1–6. IEEE (2015)
6. Barbareschi, M., Battista, E., Mazzeo, A., Mazzocca, N.: Testing 90 nm microcontroller sram puf quality. In: 2015 10th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), pp. 1–6. IEEE (2015)
7. Barbareschi, M., Battista, E., Mazzeo, A., Venkatesan, S.: Advancing wsn physical security adopting tpm-based architectures. In: 2014 IEEE 15th International Conference on Information Reuse and Integration (IRI), pp. 394–399. IEEE (2014)
8. Battista, E., Casola, V., Mazzeo, A., Mazzocca, N.: Siren: a feasible moving target defence framework for securing resource-constrained embedded nodes. Int. J. Crit. Comput.-Based Syst. **4**(4), 374–392 (2013)
9. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, p. 175. ACM (2015)
10. Böhm, C., Hofer, M., Pribyl, W.: A microcontroller sram-puf. In: 2011 5th International Conference on Network and System Security (NSS), pp. 269–273. IEEE (2011)
11. Cilardo, A., Barbareschi, M., Mazzeo, A.: Secure distribution infrastructure for hardware digital contents. CDT, IET **8**(6), 300–310 (2014)
12. Claes, M., van der Leest, V., Braeken, A.: Comparison of sram and ff puf in 65nm technology. In: Information Security Technology for Applications, pp. 47–64. Springer (2011)
13. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA Intrinsic PUFs and Their Use for IP Protection. Springer (2007)
14. Hu, W., Tan, H., Corke, P., Shih, W.C., Jha, S.: Toward trusted wireless sensor networks. ACM Trans. Sens. Netw. (TOSN) **7**(1), 5 (2010)
15. Lee, Y.S., Lee, H.J., Alasaarela, E.: Mutual authentication in wireless body sensor networks (wbsn) based on physical unclonable function (puf). In: 2013 9th International Wireless Communications and Mobile Computing, pp. 1314–1318. IEEE (2013)
16. Liu, W., Zhang, Z., Li, M., Liu, Z.: A trustworthy key generation prototype based on ddr3 puf for wireless sensor networks. Sensors **14**(7), 11542–11556 (2014)

17. Maes, R., Tuyls, P., Verbauwhede, I.: Intrinsic pufs from ip-ops on reconfigurable devices. In: Proceedings of Benelux Information and System Security, Eindhoven (2008)
18. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC press (1996)
19. Rampon, J., Perillat, R., Torres, L., Benoit, P., Di Natale, G., Barbareschi, M.: Digital right management for ip protection. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2015, pp. 200–203. IEEE (2015)
20. STMicroelectronics: RM0090 Reference Manual, 10 (2015)
21. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Proceedings of the 44th annual Design Automation Conference, pp. 9–14. ACM (2007)
22. Vatajelu, I., Di Natale, G., Barbareschi, M., Torres, L., Indaco, M., Prinetto, P.: Stt-mram-based puf architecture exploiting magnetic tunnel junction fabrication-induced variability. ACM J. Emerg. Technol. Comput. Syst. *12*(4) (2015)
23. Yang, K., Zheng, K., Guo, Y., Wei, D.: Puf-based node mutual authentication scheme for delay tolerant mobile sensor network. In: 2011 7th International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1–4. IEEE (2011)