Enabling Secure Boot Functionality by Using Physical Unclonable Functions

Kai-Uwe Müller*, Robin Ulrich*, Alexander Stanitzki*, and Rainer Kokozinski*†
*Fraunhofer IMS, 47057 Duisburg, Germany
†Universität Duisburg-Essen, Germany
Email: kai-uwe.mueller@ims.fraunhofer.de

***Abstract*—A firmware encryption for embedded devices can prevent the firmware from being read out to clone the device to a counterfeited one or to steal the intellectual property of the software developer. Also the integrity is ensured to hinder an attacker from manipulating the firmware to a malicious one. In this work, a cryptographic concept to implement a Secure Boot functionality using the intrinsic properties of a specific hardware device is shown. After describing the Physical Unclonable Function and the cipher used for the implementation, the key generation algorithm is explained. Further, the function of the crypto-module inside the system architecture and the secure boot sequence are described.**

***Index Terms*—Physical Unclonable Functions, PUF, Secure Boot, Firmware Encryption**

I. Introduction

The development of firmware is a major matter of expense in the design of embedded systems. Also a manipulation of firmware data may cause severe damage to industrial production lines or other decentralized systems. To protect the firmware from being copied and used on another hardware device or being manipulated, two general approaches can be chosen. The first is to make the firmware unreadable for an attacker, which could be achieved by encrypting the firmware with a cryptographic cipher using a hard-coded key. This would make it impossible for an attacker to read out the firmware and steal the intellectual property from the device owner if the device is powered off and the secret key is not accessible. A second approach is to associate the firmware to a specific hardware device and make it non-executable on any other one.

In this work, the two approaches are combined by using a Physical Unclonable Function (PUF) to generate a hardware-specific identifier which functions as a key for a lightweight hardware-implemented block cipher to decrypt and encrypt the firmware for providing a Secure Boot functionality for an embedded system.

II. Related Work

A common way of implementing hardware-based security in PCs and embedded systems is the trusted platform module (TPM) [1]. The standard (ISO/IEC 11889) describes a crypto-processor whose main purpose is to ensure the integrity of a platform (i.e. an embedded system). The security is based on RSA, SHA or AES using secret keys, which are saved onto the device by the manufacturer (Endorsement Key, EK) or generated internally on the device (Storage Root Key, SRK). The secret keys are not externally accessible and never have to leave the device. Unfortunately, the keys may be accessible via hardware attacks, as shown in [2] for Flash EEPROM memories. Additionally, the Endorsement Key may be saved by the manufacturer or given to federal organizations.

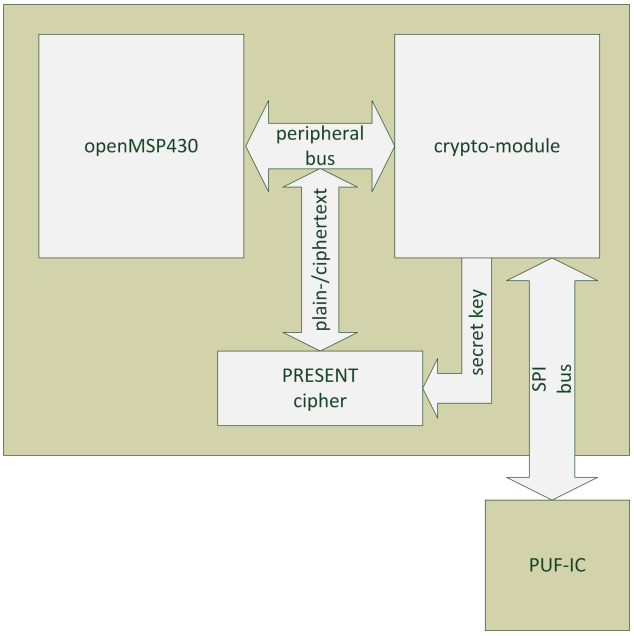

Fig. 1. System architecture

III. Embedded System Architecture

The example embedded system consists of a microcontroller device which could be connected to several sensor devices in an industrial environment. An open-source derivative of the Texas Instruments MSP430 family was used as a 16bit microcontroller, namely the openMSP430 [3]. Advantages of this design are the compatibility to the standard MSP430 toolchain and its low area usage (only 8,000 gates are needed for the implementation). It is also well suited for embedded systems because of its ultralow-power architecture to maximize the operating hours of battery powered devices. The microcontroller and the peripheral parts for key generation and

978-1-5386-5387-6/18/$31.00 ©2018 IEEE

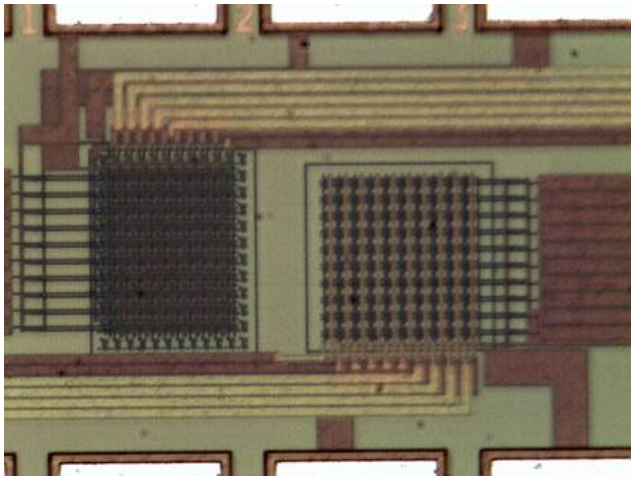

Fig. 2. Chip photograph of the PUF array

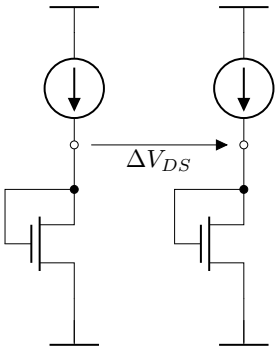

Fig. 3. PUF-pair

encryption, which are described later, are implemented on a Virtex 5 field programmable gate array (FPGA). An external IC providing the PUF functionality is connected over an SPI bus (see Fig. 1).

IV. Physical Unclonable Function

A hardware-specific identifier is created by using a Physical Unclonable Function (PUF). A PUF describes a physical one-way function that extracts intrinsic properties of an integrated circuit to output a specific sequence of bits which should be random and unpredictable, but at the same time reproducible for the exact same hardware device. For PUF, these properties can be divided into the metrics Randomness, Uniqueness, Bias and Stability [4]. For this specific implementation, the integrated differential readout-circuit for an array-based analog PUF which has been described in [5] is used.

The PUF primitives are build as diode-connected nMOS transistors with a gate-area clearly under the specified minimum dimensions of the used $350\,\mathrm{nm}$ CMOS process, namely with a size of W=$320\,\mathrm{nm}$ and L=$200\,\mathrm{nm}$, which leads to an increased mismatch of the devices. The primitives are arranged as a 16 by 16 array resulting in 256 primitives in total. A chip photograph of different array versions is shown in Fig. 2. The transistors are driven by a constant current source as shown in Fig. 3. The current-voltage characteristic of these devices varies randomly due to variations of several process parameters, such as the threshold voltage V_{th} or the transconductance, and matching errors between the individual transistors. These variations effect the drain-source voltage of the transistor. A pair building is used to form a voltage V_{DS} to only extract the matching errors instead of measuring also the overall process or environmental variations.

V. Used Cipher and Key Generation Algorithm

A lightweight cipher is needed to realize the decryption of the firmware on the embedded system. The PRESENT cipher [6] was chosen since it is a standardized lightweight block cipher in the ISO/IEC 29192-2 standard and has shown to be well implementable in hardware. The cipher needs a 128 bit key which can be exactly provided by building 128 element pairs out of the 256 elements which comprise the array of the integrated PUF circuit. The pair building is based on an approach called Sequential Pairing Algorithm (SPA), which is described briefly for the use in a Ring-Oscillator PUF implementation in [7]. In this case, the algorithm (see Algorithm 1) works as follows: at first the output voltages V_{DS} of all n elements in the array are measured. Then the elements are sorted in an descending order of the output voltages and indexed from 1 to n. From this sorted list, each element from $i = 1$ to $i = n/2$ is combined with element $j = i + n/2$ to build $n/2$ pairs if every voltage difference is greater than a specified threshold $V_{DS_{th}}$. The outputs of the devices in a pair combination now have a sufficient distance to generate a reliable bit out of the decision which one has the higher output value, i.e. whether the generated voltage ΔV_{DS} is positive or negative.

Algorithm 1 Sequential Pairing

```
1: procedure SortItems( )
2:     Sort voltages in descending order:
3:     V_DS1 > V_DS2 > ... > V_DSn
4: procedure BuildPairs
5:     i ← 1
6:     j ← n/2 + 1
7:     while i < n/2 + 1 do
8:         if V_DSi − V_DSj > V_DSth then
9:             Build Pair {Element_i, Element_j}
10:        i ← i + 1
11:        j ← j + 1
```

The pair combinations can be saved to the internal memory without leaking any information about the key or the general intrinsic properties. As an additional safety feature, the pairing list is protected by a Message Authentication Code (MAC) which also uses the generated key as a secret to hinder an adversary from manipulating the code in order to extract the secret key.

Fig. 4. Program memory organization

VI. Implementation of a crypto-module

The key generation algorithm is implemented in a separate module which also comprises an SPI Master Interface to communicate with the integrated PUF circuit. The PRESENT cipher described in section V is also implemented as a hardware module, which gets the secret key directly from the crypto-module and the encrypted data over the peripheral bus. By using a separate module to perform the key generation and decryption, the secret key is never exposed to the microcontroller. The PRESENT cipher can be seen as a black box on the peripheral bus of the microcontroller which has the ciphertext as input and outputs the plaintext.

VII. Secure Boot Sequence

The program memory is divided into two parts as shown in Fig. 4. The upper part is write-protected and contains an executable program which can perform the decryption sequence. The other part is not write-protected and contains the encrypted program and the above mentioned MAC-protected transistor pairing list. The block random-access memory (BRAM) of the FPGA is initialized with the configuration data on every startup, including the microcontroller design with an initialized program memory.

The Secure Boot Sequence is shown in Fig. 5. After powering up the system, the Decryption Program is being executed. The first step is to read the MAC_{enc}, which was generated in the encryption phase to protect the pairing list from manipulation, and the pairing list out of the header of the Critical Program. Using the information of the pairing list, the Secret Key is generated by checking if the output voltage ΔV_{DS} (see Fig. 3) for every pair P_i from $i = 1$ to $i = 128$ is positive ($bit_i = 1$) or negative ($bit_i = 0$) and concatenating the outputs to a 128bit key. The check can be done by using a simple comparator block or in this case by using the most significant bit of the analog-to-digital converter which is integrated on the PUF-IC and accessed over the SPI interface. The key is used to perform the generation of MAC_{dec} over the pairing list. MAC_{enc} and MAC_{dec} are compared to detect if the pairing list has been changed by an attacker. After the verification of the pairing list, the Critical Program is decrypted using the hardware implemented block cipher and can be executed.

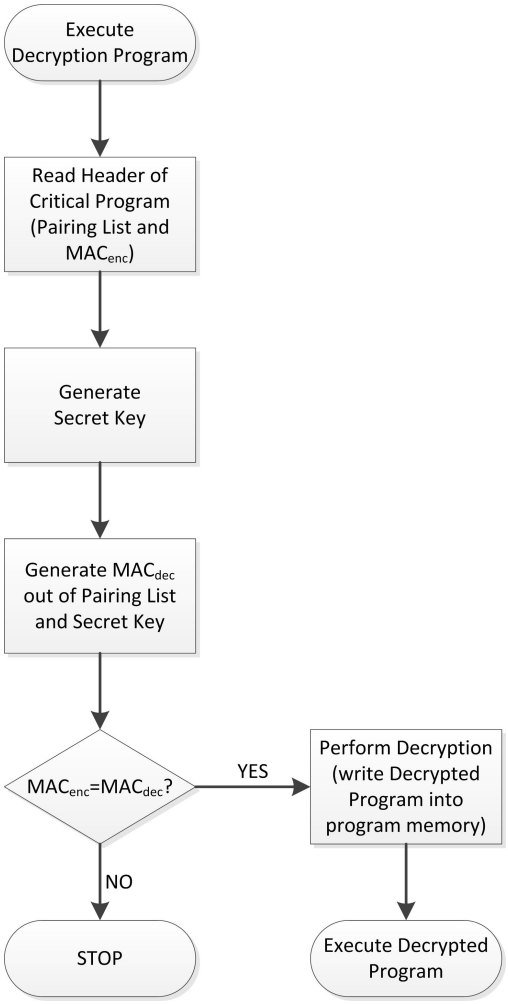

Fig. 5. Secure Boot Sequence

This sequence serves two major functions. The first is to hinder the system from executing malicious program code which changes the functionality of the device. If an attacker wants to implement his own executable program into the writable memory, it has to be encrypted with a key corresponding to a specific pairing list. This combination of key and pairing list can only be generated if the attacker knows the intrinsic properties of the device, namely the PUF responses. A possible attack to get knowledge of these properties would be to change

the pairing list to a version which implies the actual secret key. An example would be to use the same transistor pair for every key bit. This attack is prevented by adding a validation sequence which checks if every transistor pair is only used once in the key building sequence.

The second major functionality is to prevent the software from being executed on another hardware device since the secret key derived from the PUF is device-specific. This serves as an anti-counterfeiting mechanism.

VIII. Conclusion and further work

A cryptographic concept for using Physical Unclonable Functions as a hardware security anchor for a Secure Boot functionality has been developed and implemented into an FPGA. It was shown that the concept is suitable to ensure the integrity of the system and to bind a given software to a certain hardware device. A key generation algorithm to ensure a stable PUF-based key without the need of additional helper data or error correction codes was presented.

As further work the design will be implemented on an ASIC together with the PUF readout-circuit to have an embedded system in which it is impossible to get information about the PUF responses, which in this first implementation may be accessible on the SPI bus.

References

[1] TrustedComputingGroup, "Trusted platform module (tpm) summary," April 2008. [Online]. Available: https://trustedcomputinggroup.org/trusted-platform-module-tpm-summary/

[2] F. Courbon, S. Skorobogatov, and C. Woods, "Reverse engineering flash eeprom memories using scanning electron microscopy," in *CARDIS*, 2016.

[3] O. Girard, "openmsp430," June 2009. [Online]. Available: https://opencores.org/project,openmsp430

[4] M. D. Yu, R. Sowell, A. Singh, D. M'Raïhi, and S. Devadas, "Performance metrics and empirical results of a puf cryptographic key generation asic," in *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, June 2012, pp. 108–115.

[5] B. Willsch, K. U. Müller, Q. Zhang, J. Hauser, S. Dreiner, A. Stanitzki, H. Kappert, R. Kokozinski, and H. Vogt, "Implementation of an integrated differential readout circuit for transistor-based physically unclonable functions," in *2017 Austrochip Workshop on Microelectronics (Austrochip)*, Oct 2017, pp. 58–63.

[6] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, P. Paillier and I. Verbauwhede, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466.

[7] C. E. D. Yin and G. Qu, "Lisa: Maximizing ro puf's secret extraction," in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, June 2010, pp. 100–105.