# Evaluating SRAM as Source for Fingerprints and Randomness on Automotive Grade Controllers

Bogdan Groza, Pal-Stefan Murvay and Tudor Andreica
*Department of Automatics and Applied Informatics, Politehnica University of Timisoara, Timisoara, Romania*

Keywords: Physical Fingerprinting, Randomness, SRAM.

Abstract: It is well known that the state of uninitialized SRAM provides a unique pattern on each device due to physical imperfections. Both the affinity toward some fixed state as well as the deviation from it can be successfully exploited in security mechanisms. Fixed values provide an efficient mechanism for physical identification and for extracting cryptographic keys while the randomness of bits that flip can be exploited as input for PRNGs that are vital for the generation of ephemeral keys. In this work we try to give an assessment of these two capabilities on several state-of-the art automotive grade embedded platforms. The security of embedded devices inside vehicles has gained serious attention in the past years due to the impact of emerging technologies, e.g., self-driving cars, vehicle-to-vehicle communication, which are futile in the absence of the appropriate security mechanisms. Our examination of several state-of-the-art automotive grade controllers shows that SRAM can offer sufficient entropy and patterns for identification but careful testing is needed as some models fail to provide the expected results.

## 1 INTRODUCTION AND MOTIVATION

Recent experimental research (Checkoway et al., 2011), (Koscher et al., 2010) showed vehicles to be trivial targets in front of determined adversaries. This line of comprehensive academic research was preceded or followed by several other works in which a plethora of vehicular subsystems were attacked, e.g., wireless keys (Ishtiaq Roufa et al., 2010), (Verdult et al., 2012), (Wetzels, 2014), (Shoukry et al., 2013), (Tillich and Wójcik, 2012), (Francillon et al., 2011), wireless tire sensors (Verdult et al., 2012), the ABS braking system (Shoukry et al., 2013), etc. Recent surveys are done in (Miller and Valasek, 2014) and (Studnia et al., 2013).

Security tools for alleviating these problems exist, e.g., cryptographic protocols, but they do require the existence of some capabilities that may be absent in automotive grade controllers. Random number generators are one such resource that is not to be found in several automotive grade platforms, e.g., none of state-of-the-art devices that we target in the following sections has such functionality. Adding peripheral circuitry is not always an option as it increases production costs. The automotive industry is highly sensitive to financial issues as devices are usually replicated in many thousands or even millions of copies.

In terms of performance, automotive embedded devices could be compared with earlier mobile devices for which the problem of randomness generation was previously aproached (Krhovjak et al., 2009). However, the main randomness sources employed in these cases (i.e. microphone and camera input) are not generally available in automotive Electronic Control Units (ECUs).

Randomness is not the only desired functionality. The capability of uniquely identifying a device is vital as well as extracting a key that was not previously stored in the non-volatile memory. This opens road for physical identification of a device and for building cryptographic protocols that rely on keys that are harder to extract and do not require an a-priori setup. A more comprehensive discussion on such applications will follow.

The idea of using SRAM state as source for fingerprinting devices or for generating randomness was first explored by Holcomb et al. (Holcomb et al., 2007), (Holcomb et al., 2009). At the same time, it was noted by Guajardo et al. (Guajardo et al., 2007a), (Guajardo et al., 2007b), that physical unclonable characteristics can be used for assuring Intellectual Property (IP) protection, e.g., unique patterns derived from the device characteristics can be used as



Groza, B., Murvay, P-S. and Andreica, T.
Evaluating SRAM as Source for Fingerprints and Randomness on Automotive Grade Controllers.
DOI: 10.5220/0005966401090120
In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT*, pages 109-120
ISBN: 978-989-758-196-0

encryption/decryption keys making the software usable only on the device that has the prescribed characteristics. A survey on various sources behind Physically Unclonable Functions (PUFs) can be found in (Maes and Verbauwhede, 2010), another overview is available in (Rührmair and Holcomb, 2014).

While the idea of using SRAM state is not new, here we target a distinct class of application and provide experimental results on platforms that to the best of our knowledge were not subject to previous analysis, i.e., automotive grade controllers. We begin with a brief overview of potential applications in the automotive domain.

## 1.1 Target Automotive Scenarios

To emphasize on the relevance of our analysis for automotive applications, here we outline three scenarios that are target of our current and future work:

(i) *Random Number Generators.* The state-of-the-art is generous in providing sources for randomness. Oscillator drifts, the noise of Zener diodes, ring oscillators, etc. can be used as sources of entropy. However, there are many automotive embedded boards that do not have such sources for garnering entropy. Moreover, adding extra-circuitry is not always allowed and it clearly increases the production costs which is not desired by manufacturers. A relevant example is the Tire Pressure Monitoring System (TPMS) which uses miniature sensors that are placed inside the tire. Previous work (Ishtiaq Roufa et al., 2010) has shown this system to have vulnerabilities that can easily lead to several attacks: eavesdropping, vehicle tracking, battery drainage, message modification, etc. The academic community was quick to react with proposals, e.g., (Xu et al., 2013) and (Solomon and Groza, 2015), while the industry proves to be aware too (Toth, 2014). Unfortunately, the results presented in (Xu et al., 2013) are on an Arduino based platform which is not at all similar in terms of capabilities with real-world sensors (that are far more constrained). One of the most frequently used platform in TPMS systems, i.e., the Infineon SP37 from (Solomon and Groza, 2015) which is also included in our study, has no extra-circuitry for generating randomness. In the absence of a secure RNG, all the protocols proposed in (Xu et al., 2013), (Solomon and Groza, 2015) are void of practical significance. Consequently, one of the objectives of our work is to study the security of the randomness extracted from SRAM cells by providing concrete measurements. In case of the SP37 sensor, our results show that the entropy that can be extracted is not high mostly due to the small size of the available RAM.

(ii) *ECU Fingerprinting.* A common problem faced by the automotive industry is the illegal replacement of various ECUs (Electronic Control Units) inside the car. E.g., by simply replacing the BCM (Body Control Module) one can gain access to the car with a distinct car key that is paired to the newer BCM. Storing some cryptographic keys inside the non-volatile memory of the genuine ECU and later use it for authentication is not a solution in case that the adversary can gain access to this non-volatile memory and copy it to a counterfeit ECU. Cloning the software from one ECU to another one will make these ECUs indistinguishable from the software level. However, using the SRAM state as a fingerprint can be a realistic mean to decide if a particular ECU is indeed the genuine one. As part of the diagnosis systems (a standardized functionality inside a car) each ECU can be requested to externalize a unique fingerprint which can be stored by the manufacturer (or some authorized third-party) for subsequent verification. In Figure 1 we make a suggestive depiction for such a protocol. To remove the need for a more demanding protocol, we assume that the ECU registration procedure is done in a secure environment (indeed, this should be the case in practical scenarios). During the registration stage, each ECU externalizes a fingerprint, i.e., several bytes from the uninitialized SRAM, to the OBD (On-board Diagnosis Tool) which further sends them to an OEM (Original Equipment Manufacturer) server where the fingerprint is stored. To make the ECU cooperate and externalize the fingerprint, the OBD tool first sends a request req which is signed by the OEM along with a time-stamp $t_{req}$ to enforce freshness. The signature of the OEM $s_{OEM} = Sign_{OEM}(req, t_{req})$ is used to enforce only authorized sessions to acquire this fingerprint. The ECU then externalizes the fingerprint $\mathscr{F}_{ECU}$ along with its identifier $id_{ECU}$ which are signed by the OBD tool and sent to the OEM server. Later, for ECU identification, we rely on *fuzzy cryptography* as the fingerprint extracted from the ECU may present small variations when compared to the original one. First, the ECU sends its identification number $id_{ECU}$ and some random value $rnd_{ECU}$. Then

the OBD tool sends a value that is encrypted with the fingerprint $\text{enc}_{\mathscr{F}_{\text{ECU}}}(\text{rnd}_{\text{ECU}}, \text{rnd}_{\text{OBD}})$ and the helper data $\mathscr{H}_{\mathscr{F}_{\text{ECU}}}$ which will be used to compensate for the noise in the new fingerprint. The ECU proves its identity by successfully decrypting the random value $\text{rnd}_{\text{OBD}}$. The encrypted challenge $\text{enc}_{\mathscr{F}_{\text{ECU}}}(\text{rnd}_{\text{ECU}}, \text{rnd}_{\text{OBD}})$ can be of course generated by either the OBD tool by acquiring the fingerprint $\mathscr{F}_{\text{ECU}}$ from the OEM or by the OEM itself.

ECU registration (Secure Environment)

1. OBD $\rightarrow$ ECU: $\text{req}, \text{t}_{\text{req}}, \text{s}_{\text{OEM}} = \mathit{Sign}_{\text{OEM}}(\text{req}, \text{t}_{\text{req}})$
2. ECU $\rightarrow$ OBD: $\text{id}_{\text{ECU}}, \mathscr{F}_{\text{ECU}}$
3. OBD $\rightarrow$ OEM: $\text{id}_{\text{ECU}}, \mathscr{F}_{dev}, \text{s}_{\text{OBD}} = \mathit{Sign}_{\text{OBD}}(\text{id}_{\text{ECU}}, \mathscr{F}_{dev})$

ECU identification (Insecure Environment)

1. ECU $\rightarrow$ OBD: $\text{id}_{\text{ECU}}, \text{rnd}_{\text{ECU}}$
2. OBD $\rightarrow$ ECU: $\mathscr{H}_{\mathscr{F}_{\text{ECU}}}, \text{enc}_{\mathscr{F}_{\text{ECU}}}(\text{rnd}_{\text{ECU}}, \text{rnd}_{\text{OBD}})$
3. ECU $\rightarrow$ OBD: $\text{rnd}_{\text{ECU}}, \text{rnd}_{\text{OBD}}$

Figure 1: Suggestive depiction of component registration and subsequent identification based on fingerprint.

(iii) *Generation of unique cryptographic keys.* There are numerous devices inside the car which need to operate with a unique cryptographic key. A good example are wireless car keys (RF keys) which despite being produced in thousands of identical copies need to be seeded with unique cryptographic keys (to avoid impersonation). Many other examples can be envisioned, but car keys are one of the most intensively studied vehicular device both in terms of practical instances, e.g., (Verdult et al., 2012), (Wetzels, 2014), (Shoukry et al., 2013), (Tillich and Wójcik, 2012), (Francillon et al., 2011) or cryptographic designs (Biham et al., 2008). Using memory patterns offers an excellent alternative to build an per-item unique key without specific programming of the device, i.e., the uniqueness of the key is simply guaranteed by the uniqueness of the memory pattern. In Figure 2 we make a suggestive depiction for a protocol intended for key registration and subsequent access to the car, this procedure is of course similar to the ECU fingerprinting. First an OBD tool initializes the BCM (Body Control Module) to accept the fingerprint from an RF key $\mathscr{F}_{dev}$. The fingerprint is sent by the RF key along with its identification number $\text{id}_{\text{RFKey}}$. Again, to remove the need for a more demanding protocol, we assume that the key registration procedure is done in a secure environment. For subsequent authentication, the BCM sends a random challenge $\text{rnd}_{\text{BCM}}$ which is coupled with the challenge from the key $\text{rnd}_{\text{RFKey}}$ and encrypted with the fingerprint $\mathscr{F}'_{dev}$ (note that $\mathscr{F}'_{dev}$ may have some bits affected by noise in comparison with the original $\mathscr{F}_{dev}$) as $\text{enc}_{\mathscr{F}'_{dev}}(\text{rnd}_{\text{RFKey}}, \text{rnd}_{\text{BCM}})$. For successful decryption and verification from the BCM, this is send along with helper data $\mathscr{H}_{\mathscr{F}_{dev}}$.

Key registration (Secure Environment)

1. OBD $\rightarrow$ BCM: $\text{req}, \text{t}_0, \mathit{Sign}_{\text{OEM}}(\text{req}, \text{t}_0)$
2. RFKey $\rightarrow$ BCM: $\text{id}_{\text{RFKey}}, \mathscr{F}_{dev}$

Authentication (Insecure Environment)

1. RFKey $\rightarrow$ BCM: $\text{rnd}_{\text{RFKey}}$
2. BCM $\rightarrow$ RFKey: $\text{rnd}_{\text{BCM}}$
3. RFKey $\rightarrow$ BCM: $\mathscr{H}_{\mathscr{F}_{dev}}, \text{enc}_{\mathscr{F}'_{dev}}(\text{rnd}_{\text{RFKey}}, \text{rnd}_{\text{BCM}})$

Figure 2: Suggestive depiction of key registration and subsequent authentication based on fingerprint.

These applications are graphically suggested in Figure 3. The protocol depictions in Figures 1 and 2 are mere suggestions as it should be clear that practical adoption demands both standardization and a more rigorous security analysis. But before that, in order to test the feasibility of these ideas we need some experimental measurements for the quality of SRAM in providing the desired entropy and unique patterns. This is the main goal of our work.

Our work is organized as follows. In Section 2 we provide a brief overview on the automotive grade controllers that we put to test. Section 3 provides an overview of the mathematical tools that we use for the evaluation while section 4 provides the experimental data.

## 2 DEVICES AND TECHNICAL DETAILS

We begin by presenting the devices that are target of our experiments, these are specifically chosen to meet a large variety of automotive applications. Then we proceed to presenting some technical details on how memory dumps were performed on each of the devices.

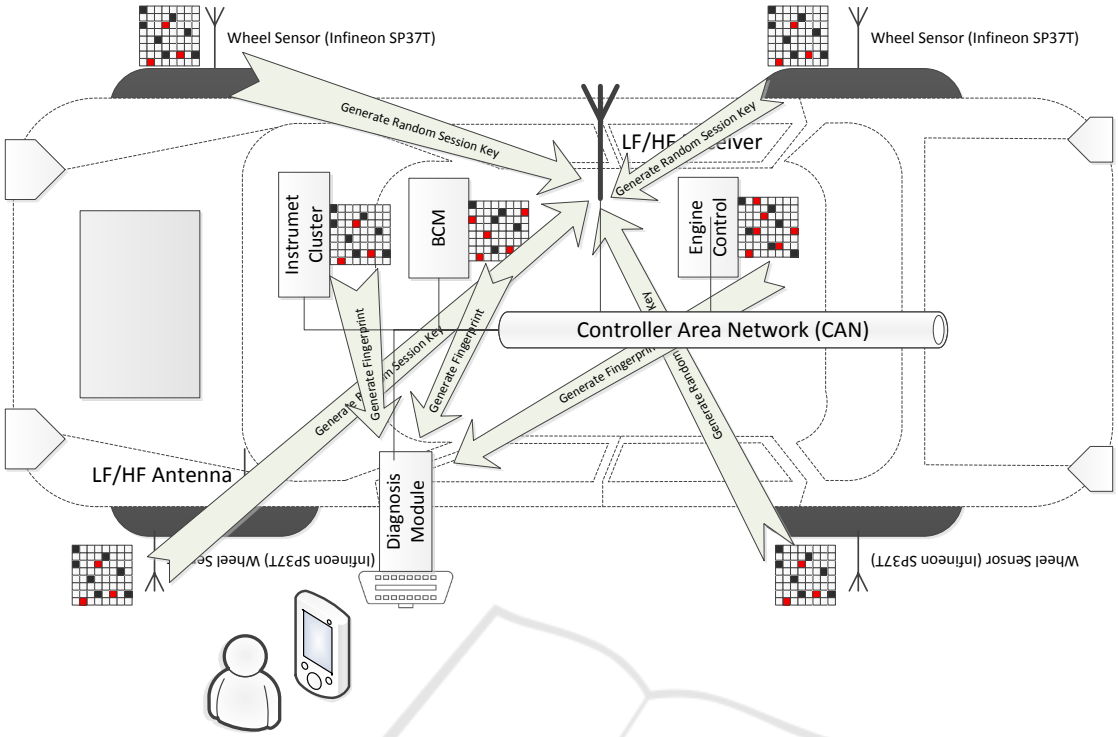



Figure 3: Suggestive depiction of our target automotive applications: generating randomness and ECU fingerprinting (memory cells depicted as constant zeros (white), ones (black) or flipping (red).

## 2.1 Target Devices

The chosen microcontrollers cover the major device categories employed in the automotive industry in terms of available resources and processing power. We now enumerate through these platforms and also give hints on the target automotive scenario for each of them:

(i) INFINEON SP37. This device represents the low-end of the hierarchy, an application specific integrated circuit built for tire pressure sensing around an 8-bit 8051 core. The *SP37* sensor has a RAM memory that is limited to 256 bytes. Due to inherent constraints on this platform both fingerprinting and generating randomness are important functionalities for the applications where this sensor is deployed.

(ii) FREESCALE S12 FAMILY. From the 16-bit category we selected three members of the S12 microcontroller family: *MC9S12C128* , *MC9S12DT256* and *MC9S12XDP512* . They are all built on the same S12 architecture with the last one having an additional coprocessor called XGATE. In terms of available RAM the three platforms have 4KB, 12KB and respectively 40KB of RAM. Members of this family are suited for vehicle body components with average performance needs.

(iii) TEXAS INSTRUMENTS MSP FAMILY. Another 16-bit candidate included in our study is the *MSP430F2274* from Texas Instruments which can be used in various vehicle body and infotainment applications. While members of the MSP430 family can have up to 8KB of RAM, our device features a memory of 1KB.

(iv) RENESAS RL78 FAMILY. The RL78/F14 is a 16-bit microcontroller from Renesas. It is used in various automotive applications such as motor and body control, etc. The R5F10PPJ core that we tested is equipped with a 20KB RAM memory.

(v) INFINEON TRICORE FAMILY. In the high-end 32-bit microcontroller area we looked at the Infineon TriCore family and selected two devices, *TC1782* and *TC1797* designed for demanding applications in the powertrain domain. They both integrate the same TriCore 1.3.1 core using a total of 176KB of RAM and an additional coprocessor for peripheral management with 16KB of RAM. The total RAM available is comprised of several sections of memory with various roles. We only used dumps

from those RAM areas which are not used as caches. The RAM associated with the main processor is called Local Data RAM (LDRAM) and has a size of 124KB that could be increased to 128KB if the data cache is not used. The main core makes use of a program memory interface with a 24KB Scratch-Pad RAM (SPRAM) which could go up to 40KB if reducing the instruction cache size. The program memory unit holds an additional 8KB of RAM called Overlay RAM. Finally, the 16KB of RAM associated with the coprocessor is called Parameter RAM (PRAM). *TC1782* features ECC (Error-Correcting Code) protection for the RAM memories. Therefore, it is necessary to have the RAM initialised to get a reliable working ECC. If the initial state of the ECC-protected memory is undefined the system will report and try to correct bit errors that are erroneously detected.

(vi) OTHER PLATFORMS. Two additional platforms from Freescale were also part of our study: S12ZVHY64 and MPC5606B. They too have ECC protected RAM but due to the way in which the error protection mechanism is implemented, the RAM on these platforms is not suitable for fingerprinting or as randomness sources. On the S12ZVHY64, the first access to an uninitialised RAM location leads to the initialization of that area before the read data is returned. In the case of the MPC5606B a read from the uninitialised RAM generates an error interrupt which leads to a reset making it impossible to read the actual random data from the memory. These are just two examples of devices on which ECC protection prevents the access to the raw RAM data after power-on reset but similar situations can be encountered on various other platforms.

Table 1 briefs over the employed platforms by also specifying the size of available RAM on each.

## 2.2 Technical Details

We used two approaches for reading out the RAM content from the selected microcontrollers. The first one is based on the debugger interface while the second one uses serial communication.

The debugger interface was used for all platforms where the corresponding debugger software provided features for saving the content of the entire memory areas. For this, a debugger connection was established following a hard-reset with or without downloading of object code (program download to flash was avoided where RAM areas were used as buffers during the flashing process). Program execution was stopped at the first instruction that allowed breakpoint setting and at this point the content of RAM was read and saved for processing. The applications residing on the microcontrollers were in all cases limited to just the most basic system initialization. This approach was used for the MSP430 and TriCore platforms.

Some debugger software products do not offer the possibility of dumping the content of memory areas to a file. This was the case for the debuggers used for the S12 and SP37 platforms. To address this, we implemented a small application that read the content of the RAM and sent it through a serial interface to a PC on which the dump files were built. This compromised the initial content of several bytes in RAM which were needed for program execution. The compromised memory locations were not included in the analysis.

For each of the devices in our study, 100 memory dumps were taken for the analysis. Increasing the number of samples from 50 to 100 did not lead to significant changes in the results, thus we considered 100 samples to be a sufficient amount.

# 3 TESTING METHODOLOGY AND RESULTS

We begin by a brief overview of the mathematical tools behind our evaluation. We then proceed to the measurements achieved for each of the devices.

## 3.1 Mathematical Tools

We now give a formal treatment of the mathematical tools that we use in our quantitative assessment of SRAM for providing fingerprints and randomness. We denote as fingerprint $\mathscr{F}_{dev}$ of device *dev*, one fixed memory dump of the device which is the set of all values $m$ for the $\ell$ memory locations taken into account , i.e.,

$$\mathscr{F}_{dev} = \{m_0^{dev}, m_0^{dev}, ..., m_\ell^{dev}\}$$

We do define the memory state $S_{dev}^i$ of device *dev* in each subsequent experiment $i \in [1..e]$ as:

$$S_{dev}^i = \{m_0^{dev}, m_0^{dev}, ..., m_\ell^{dev}\}$$

Over all the experiments space, for any pair of devices $dev', dev''$ we can now define two random variables which are of relevance to our analysis: the *intra-distance* between distinct memory mappings of the same device and the *inter-distance* between memory mappings of two devices. That is,

Table 1: Automotive grade platforms subject of our study.

| Device | Architecture | Frequency | RAM size | Manufacturer |
|---|---|---|---|---|
| *SP37* | 8 bit | 12MHz | 256B | Infineon |
| *MC9S12C128* | 16 bit | 50 MHz | 4KB | Freescale |
| *MC9S12DT256* | 16 bit | 50 MHz | 12KB | Freescale |
| *MC9S12XDP512* | 16 bit | 80 MHz | 20KB | Freescale |
| *MSP430F2274* | 16 bit | 16 MHz | 1KB | Texas Instruments |
| *RL78/F14* | 32 bit | 64 MHz | 20KB | Renesas |
| *TC1782* | 32 bit | 180 MHz | 176KB | Infineon |
| *TC1797* | 32 bit | 180 MHz | 176KB | Infineon |

(i) the *intra-distance* is the Hamming distance between memory mappings of device $dev'$ and the fingerprint $\mathscr{F}_{dev'}$ of device $dev'$, defined as a random variable over all $e$ experiments:

$$I_{dev'} = \text{Hamming}(\mathscr{F}_{dev'}, \text{S}^i_{dev'}), \forall i \in [1..e]$$

(ii) the *inter-distance* is the Hamming distance between the memory mappings of device $dev''$ and the fingerprint $\mathscr{F}_{dev'}$ of device $dev'$, defined as a random variable over all $e$ experiments:

$$\mathscr{E}_{dev',dev''} = \text{Hamming}(\mathscr{F}_{dev'}, \text{S}^i_{dev''}), \forall i \in [1..e]$$

The *intra-distance* and *inter-distance* are useful measurements for establishing the quality of fingerprints that can be extracted from the device.

For assessing the quality of random sequences and the entropy that results we need to measure the guessing probability and entropy for each cell. These are defined as follows:

(iii) let memory cell $m_i$ (taken as byte) of device *dev* be a random variable over the experiment space, the *guessing probability* of memory cell $m_i$ on device *dev* is defined as the maximum probability that cell $m_i$ takes a particular value, i.e.,

$$\gamma_{dev}(m_i) = \max\{\Pr[m_i = v] : \ v \in \{0..255\}\}^1$$

(iv) by definition, the *minimum entropy* of memory cell $m_i$ is $\log_2(1/\gamma_{dev}(m_i))$ and consequently we define *the minimum entropy* that can be extracted from the memory of device *dev* as the sum of entropies for all the memory cells, i.e.,

$$\text{H}_{dev} = \sum_{i=1,\ell} \log_2(1/\gamma_{dev}(m_i))$$

To account for potential manufacturing problems between devices of the same type, we do measure the Hamming distance between adjacent cells as well as the distance between cells of extreme behaviour. For the later, we take into account both bits of low entropy and high entropy, where by low or high we define bits that are greater or equal in value to one standard deviation from the mean.

Let $\gamma_{dev}$ be the random variable that denotes the guessing probability of an arbitrary memory cell from device *dev*. Having the mean $\mu(\gamma_{dev})$ and variance $\text{Var}(\gamma_{dev})$, let the following two set of indexes defined over the guessing probabilities of each cell of device *dev*:

$$\mathbb{I}^{\text{low}}_{dev} = \left\{ i : \gamma(m^{dev}_i) \leq \mu(\gamma_{dev}) + \text{Var}(\gamma_{dev}) \right\}$$

$$\mathbb{I}^{\text{high}}_{dev} = \left\{ i : \gamma(m^{dev}_i) \geq \mu(\gamma_{dev}) + \text{Var}(\gamma_{dev}) \right\}$$

Let, $\lambda, \Lambda$ denote the cardinalities of the sets defined above $\mathbb{I}^{\text{low}}_{dev}$ and $\mathbb{I}^{\text{high}}_{dev}$, then:

(v) we define the set of successive distances between *locations of extreme cells* with low or high entropy on device *dev* as:

$$\mathcal{L}^{\text{low}}_{dev} = \left\{ d : d \leftarrow \min\left(|x_{i-1} - x_i|, |x_i - x_{i+1}|\right), \ i = 2..\lambda{-}1, x_i \in \mathbb{I}^{\text{low}}_{dev} \right\}$$

$$\mathcal{L}^{\text{high}}_{dev} = \left\{ d : d \leftarrow \min\left(|x_{i-1} - x_i|, |x_i - x_{i+1}|\right), \ i = 2..\Lambda{-}1, x_i \in \mathbb{I}^{\text{high}}_{dev} \right\}$$

(vi) we define the set of *Hamming distances* between pairs of adjacent cells on device *dev* as:

$$\mathcal{H}_{dev} = \left\{ d : d \leftarrow \text{Hamming}(m^{dev}_{2*i}, m^{dev}_{2*i+1}), \ \forall i \in [1..\ell/2] \right\}$$

The *Hamming distances* between pairs of adjacent cells, while trivial to define, proved to be a very relevant metric as at least two devices that yield good entropy values later failed on this metric. Thus the metric proves to be a very good sanity check.

[1]The equation considers 8-bit memory cells.

## 3.2 Experimental Data

The experimental results are obtained by analyzing 100 consecutive memory dumps on each of the devices. For fingerprints, we used the first 64, 128 and 256 bits from the memory. By taking bytes located at the end of the memory the results remained the same as taking bytes from the beginning. The rest of the analysis, i.e., guessing probability and entropy, was done on the full memory dump over the entire set of 100 experiments.

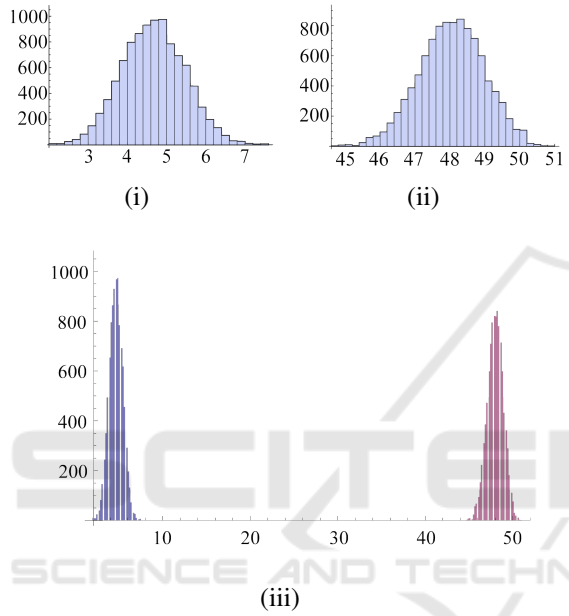



Figure 4: Experimental results on an *MC9S12XDP512* : intra-distance (i), inter-distance (ii) and the two combined (iii) - measured as % for a larger fingerprint of 512 bits.

FINGERPRINTING. The results were satisfactory on most of the devices, with isolated abnormalities on which we defer the discussion for the forthcoming paragraph. The complete experimental results are shown in Table 2. For fingerprinting, memory dumps from the same device show a variation for the mean of the intra-distance $\mu(I_{dev'})$ from 2.35% to 6.36% on all of the Freescale S12 cores. When compared between distinct devices the mean inter-distance $\mu(\mathcal{E}_{dev',dev''})$ ranged from 45.12% up to 51.73% which shows a clear cut between the intra and inter-distances. This allows an exact identification of the device from a small fingerprint, e.g., 64-256 bit. Figure 4 shows the plotted histogram for an *MC9S12XDP512* core.

FINGERPRINT OVERLAPS FOR *MSP430F2274* . On the Texas Instruments platforms the results were far from being satisfactory. The overlap between the fingerprint from one device and another is high. The mean of the intra-distance $\mu(I_{dev'})$ is between 7.28%–8.81% and the mean of the inter-distance $\mu(\mathcal{E}_{dev',dev''})$ between 31.90%–36.96% which shows that fingerprints are statistically distinguishable. However, the high variance which leads to clear overlaps suggests that multiple memory dumps will be needed for correct identification. We are left without a clear explanation for the source of this overlap and we can imagine that distinct development boards equipped with these cores may lead to distinct results. But for the devices that we worked with, we are skeptical on their potential practical use for this purpose. Figure 5 shows such an overlap for a 512 bit fingerprint, which should be sufficiently large for practical demands but still cannot lead to a clean cut between the intra and inter-distance.

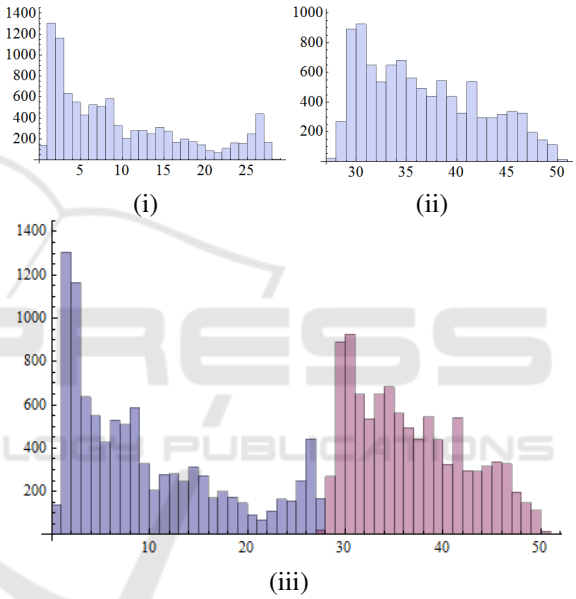



Figure 5: Experimental results on *MSP430F2274* : intra-distance (i), inter-distance (ii) and the two combined (iii) - measured as % for a fingerprint of 512 bits.

```
f6 fb 62 ff bf de fc 44     1b 18 10 20 20 60 80 2
ff f7 eb d7 ff fe f7 bf     0 0 80 1  0 15 91 68
37 ff f1 ff ff c7 5e bf     97  0 a0 1 a0 4 11 24
6d f6 65 af ff 6f bf eb     0 0 82 54 81 1 3 a2
```

Figure 6: Biased chunks from un-initialized SRAM: 64 bits at odd addresses (left) and 64 bits at even (right) addresses on TC1782.

RANDOMNESS. The complete experimental results related to entropy are shown in Table 3 which summarizes the mean and variance for the guessing probability of each cell $\gamma_{dev}$, the entropy of each cell $H_{dev}$ and the hamming distances of adjacent cells $\mathcal{H}_{dev}$. Graphical depictions are provided for the state of SRAM on each of the devices: for *MC9S12C128* and *MC9S12DT256* in Figure 8,

Table 2: Determined values for intra and inter-distances on the experimental devices (rows containing atypical results are highlighted in gray).

| Device | Fingerprint Size | Intra-distance | | Inter-distance | |
|---|---|---|---|---|---|
| | | Mean | Variance | Mean | Variance |
| | $\lvert\mathcal{F}_{dev}\rvert$ | $\mu(I_{dev'})$ | $\mathrm{Var}(I_{dev'})$ | $\mu(\mathcal{E}_{dev',dev''})$ | $\mathrm{Var}(\mathcal{E}_{dev',dev''})$ |
| *MC9S12XDP512* | 64 bit | 5.02% | 5.15% | 46.93% | 7.72% |
| *MC9S12XDP512* | 128 bit | 5.61% | 2.66% | 45.40% | 4.11% |
| *MC9S12XDP512* | 256 bit | 4.87% | 1.12% | 49.08% | 1.78% |
| *MC9S12DT256* | 64 bit | 2.35% | 2.52% | 48.74% | 5.57% |
| *MC9S12DT256* | 128 bit | 3.86% | 2.33% | 47.26% | 3.36% |
| *MC9S12DT256* | 256 bit | 3.71% | 1.10% | 48.95% | 1.64% |
| *MC9S12C128* | 64 bit | 4.95% | 5.43% | 45.12% | 4.59% |
| *MC9S12C128* | 128 bit | 6.36% | 5.04% | 51.73% | 2.85% |
| *MC9S12C128* | 256 bit | 5.35% | 2.48% | 47.91% | 1.81% |
| *RL78/F14* | 64 bit | 5.36% | 13.88% | 46.40% | 18.22% |
| *RL78/F14* | 128 bit | 5.62% | 6.85% | 50.25% | 6.82% |
| *RL78/F14* | 256 bit | 5.64% | 4.90% | 48.53% | 2.89% |
| *MSP430F2274* | 64 bit | 8.81% | 123.57% | 31.90% | 81.35% |
| *MSP430F2274* | 128 bit | 7.28% | 83.83% | 35.21% | 49.38% |
| *MSP430F2274* | 256 bit | 7.72% | 72.91% | 36.96% | 39.25% |

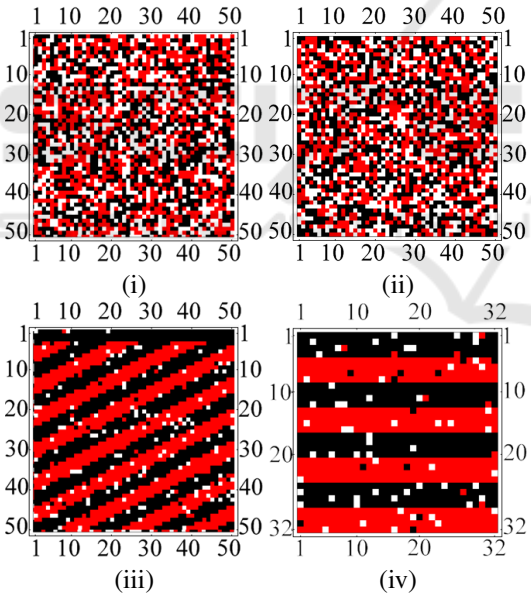



Figure 7: Distribution of bytes with dominant ones (red), zeros (black) or half zeros half ones (white) for: (i) *MC9S12XDP512* , (ii) *TC1797* , (iii) *TC1782* and (iv) *MSP430F2274* .

*MSP430F2274* in Figure 9, *SP37* in Figure 10, for the 4 memory areas from *TC1782* and *TC1797* in Figures 11, 12, 13 and 14. In all of the previous figures, we depict the guessing probability of consecutive memory cells in the upper left corner followed by the histogram for the distribution of guessing probabilities on the right side. Then we depict the entropy of consecutive memory cells in the bottom left corner and the histogram for the distribution of entropies on the right side. All of the plots show similar distributions. The entropy varied from 0.39 bits/byte in case of the *MC9S12C128* core up to 0.70bits/byte in case of the *TC1797* core. This amount of entropy should be enough for devices equipped with sufficient memory. However, for the case of the *SP37* the results show that the amount of entropy that can be collected is at 0.44 bits/byte leading to a total of around 53 bits of entropy (assuming un-initialized RAM). This is likely insufficient for generating cryptographic keys, fortunately the other environmental data that sensor collects can be used (acceleration, temperature, pressure, etc.) and this may lead to sufficient entropy. Thus the practical use of SRAM along with other sources remains and alternative for the *SP37* core. Due to several abnormalities, the case of the *TC1782* and *MSP430F2274* cores are separately discussed below. With the exception of the previous two devices, the hamming distance of adjacent cells is close to that of random cells, having the mean at around 4 bits. These distances are shown in the last two columns of Table 3.

BIAS ON *TC1782* AND *MSP430F2274* . A rather unexpected result was achieved on Infineon *TC1782* . While the memory from *TC1782* is of ECC type, we expect that the entropy would be close to 0. This happened with the PRAM, SPRAM and OVRAM. However, for the LDRAM the entropy was higher than for all of the other devices at 1.02 bits/byte.

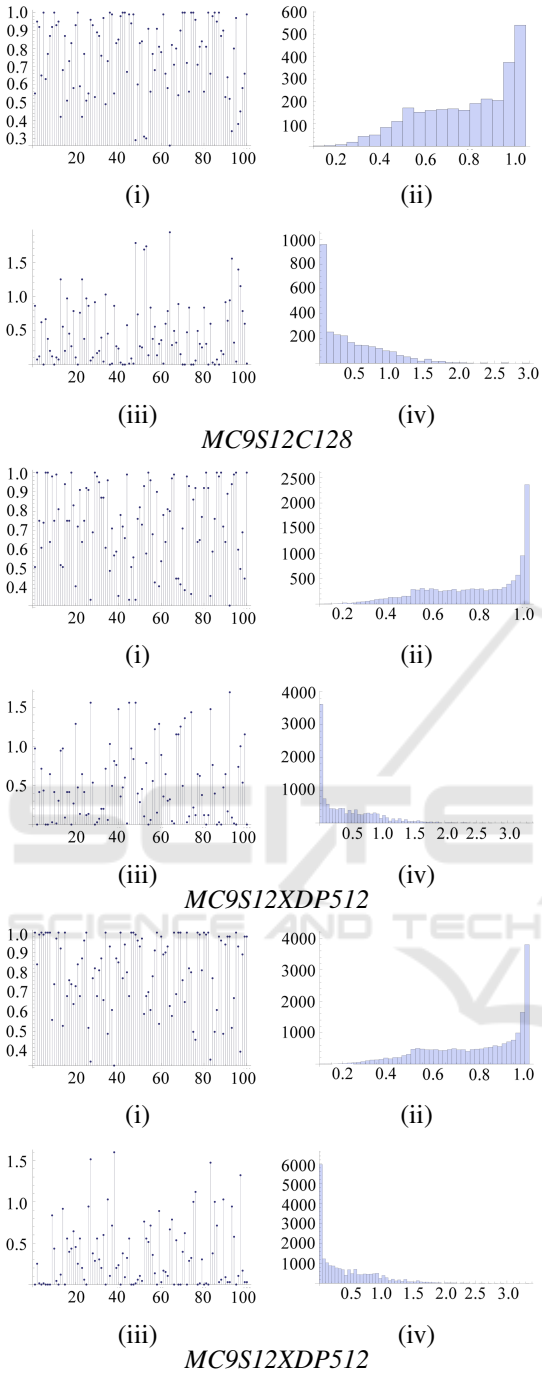



Figure 8: Experimental results on the three controllers from Freescale: guessing probability (i), guessing probability distribution (ii) entropy (iii) and entropy distribution (iv).

This finding is however contrasted by the bit patterns that could be found in the memory. Notably, bits from 64-bit blocks at odd addresses showed a bias toward 1, while bits in 64-bit blocks at even addresses showed a bias toward 0. Figure 6 shows parts of a memory dump and the bias is clearly visible. This

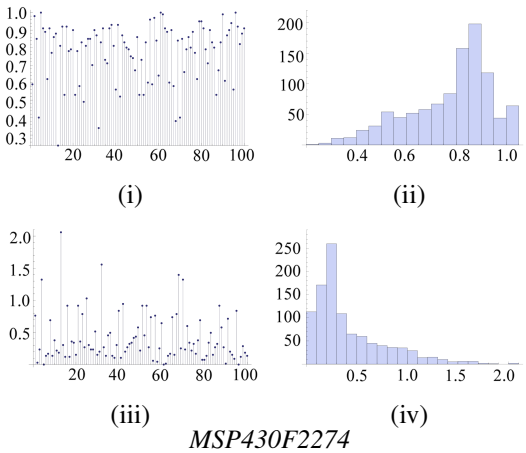



Figure 9: Experimental results on TI RF2500: guessing probability (i), guessing probability distribution (ii) entropy (iii) and entropy distribution (iv).

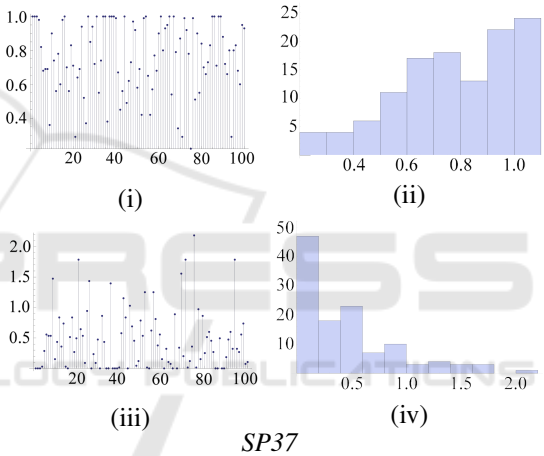
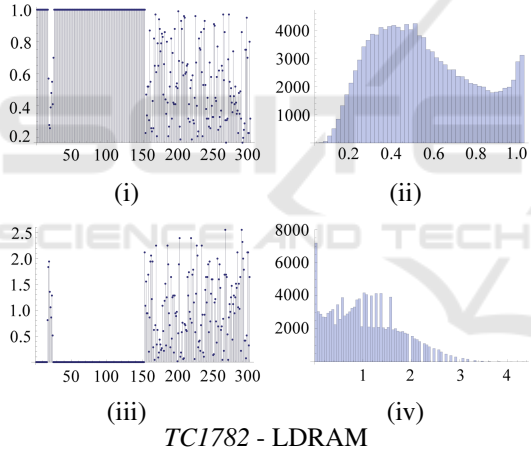



Figure 10: Experimental results on *SP37* : guessing probability (i), guessing probability distribution (ii) entropy (iii) and entropy distribution (iv).

shows that for the ECC memory on *TC1782* , despite the fact that bits do frequently changed their state, there is a correlation between adjacent locations and thus there exists dependence between bytes. For *MSP430F2274* a similar bias was observed, however, according to the datasheet of the product it is not equipped with ECC memory. A visual inspection of the memory dumps showed large portions that are biased toward 1 followed by portions that are biased toward 0. Figure 7 depicts these distributions for *TC1782* and *MSP430F2274* in contrast with two cores that do appear to have random behaviour: *MC9S12XDP512* and *TC1797* . Bytes with dominant ones are colored in red, dominant zeros are black while half-zero/half-one bytes are white. The patterns are clearly not random. For the moment we are unable to find any explanation for this behaviour, but it will stay in our focus for future work. We are how-

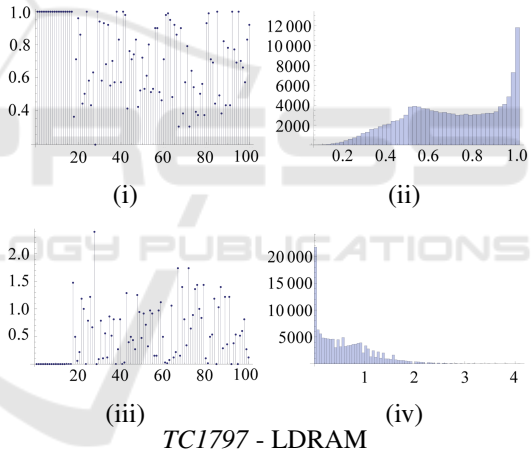
Table 3: Determined values for guessing probability, entropy and Hamming distances of adjacent cells on the experimental devices (rows containing atypical results are highlighted in gray).

| Device | Guessing (probability/byte) | | Entropy (bits/byte) | | Hamming (bits) | |
|---|---|---|---|---|---|---|
| | Mean | Variance | Mean | Variance | Adjacent | Random |
| | $\mu(\gamma_{dev})$ | $\mathrm{Var}(\gamma_{dev})$ | $\mu(\mathrm{H}_{dev})$ | $\mathrm{Var}(\mathrm{H}_{dev})$ | $\mu(\mathcal{H}_{dev})$ | $\mu(\mathcal{H}_{dev})$ |
| *MC9S12XDP512* | 0.78 | 0.04 | 0.41 | 0.20 | 3.98 | 4.00 |
| *MC9S12DT256* | 0.79 | 0.04 | 0.39 | 0.20 | 3.96 | 3.98 |
| *MC9S12C128* | 0.77 | 0.04 | 0.42 | 0.20 | 3.98 | 3.98 |
| *MSP430F2274* | 0.76 | 0.02 | 0.41 | 0.13 | 2.75 | 3.99 |
| *SP37* | 0.76 | 0.04 | 0.44 | 0.22 | 3.75 | 3.86 |
| *RL78/F14* | 0.73 | 0.04 | 0.51 | 0.23 | 3.62 | 4.00 |
| *TC1782* LDRAM | 0.54 | 0.05 | 1.02 | 0.46 | 2.58 | 3.98 |
| *TC1782* PRAM | 0.90 | 0.02 | 0.16 | 0.09 | 0.86 | 3.95 |
| *TC1782* SPRAM | 0.90 | 0.02 | 0.17 | 0.10 | 0.93 | 4.03 |
| *TC1782* OVRAM | 0.96 | 0.01 | 0.06 | 0.03 | 0.41 | 3.98 |
| *TC1797* LDRAM | 0.70 | 0.04 | 0.59 | 0.28 | 3.93 | 3.99 |
| *TC1797* PRAM | 0.65 | 0.05 | 0.70 | 0.34 | 3.89 | 3.98 |
| *TC1797* SPRAM | 0.66 | 0.05 | 0.69 | 0.33 | 3.93 | 4.00 |
| *TC1797* OVRAM | 0.68 | 0.05 | 0.64 | 0.30 | 3.87 | 3.99 |




*TC1782* - LDRAM

Figure 11: Null entropy cells (due to preloaded values) followed by average entropy cells on the LDRAM of TriCore TC1782: guessing probability (i), guessing probability distribution (ii) entropy (iii) and entropy distribution (iv).



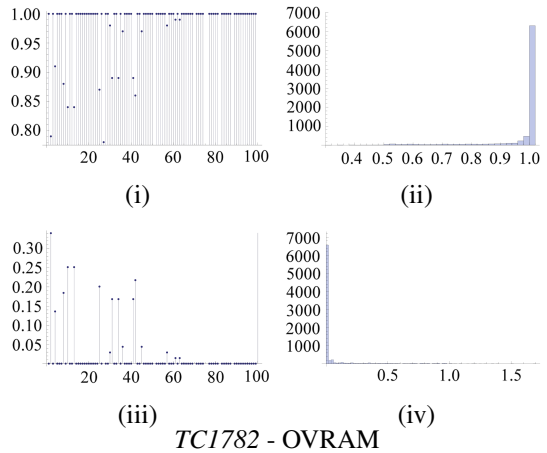

*TC1797* - LDRAM

Figure 12: Constant entropy cells (due to preloaded code) followed by average entropy cells on the LDRAM of Infineon TriCore TC1797: guessing probability (i), guessing probability distribution (ii) entropy (iii) and entropy distribution (iv).

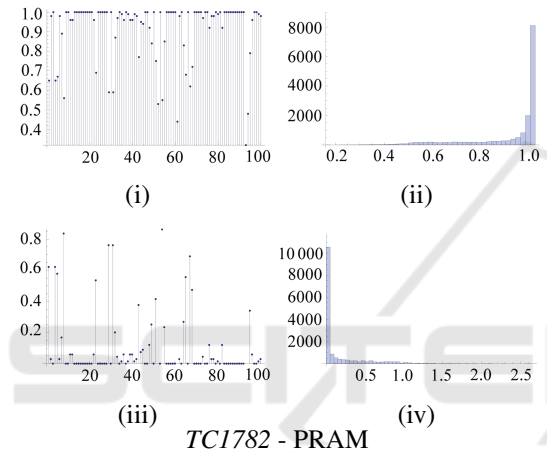
ever pleased by the fact that the Hamming distance of adjacent cells proved to be a marker that quickly predicted this behavior. The result for the Hamming distances of adjacent cells vs. randomly selected cells are also depicted in Table 3. The distances between cells of extreme values showed no abnormalities on any of the devices, for this reason the experimental values are not included.

# 4 CONCLUSION

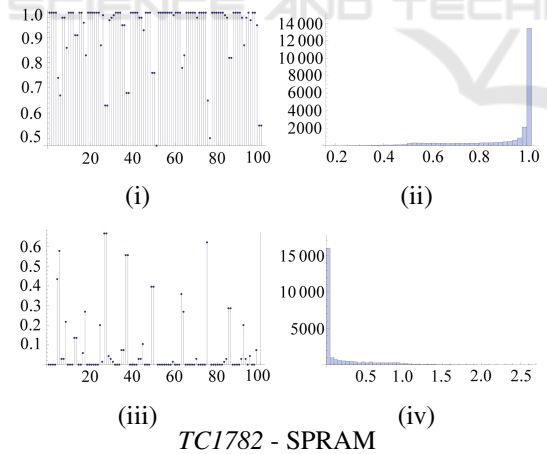
Our work provides comprehensive results on the quality of SRAM for providing fingerprints and randomness on automotive grade controllers. The results show that SRAM can be considered a good source for both randomness and fingerprints but careful testing is needed as the results are not uniform between devices. Notable, the *MSP430F2274* controllers that we used do not appear to provide enough accuracy




*TC1782* - OVRAM

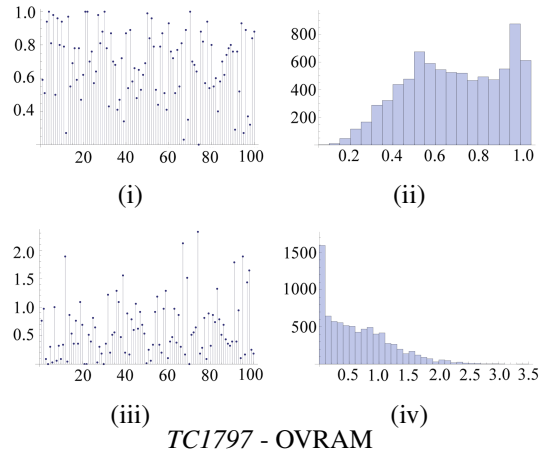



*TC1782* - PRAM

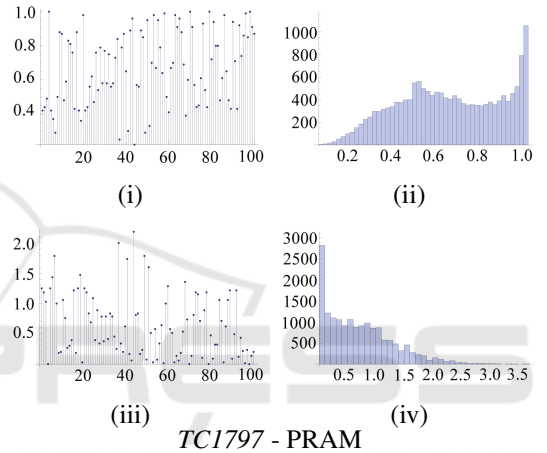



*TC1782* - SPRAM

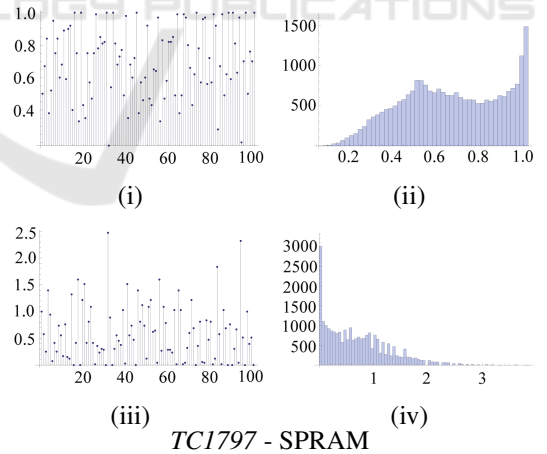
Figure 13: High guessing probability and lack of entropy for OVRAM, PRAM and SPRAM on Infineon TriCore TC1782: guessing probability (i), guessing probability distribution (ii) entropy (iii) and entropy distribution (iv).




*TC1797* - OVRAM




*TC1797* - PRAM




*TC1797* - SPRAM

Figure 14: Regular values for entropy on OVRAM, PRAM and SPRAm for Infineon TriCore TC1797: guessing probability (i), guessing probability distribution (ii) entropy (iii) and entropy distribution (iv).

for fingerprinting. As expected, the ECC memory of devices such as the *TC1782* controller is not a good source for randomness. In particular for *TC1782* controller the entropy of certain portions of RAM was high when strictly drawn from the state of each byte after reset, but there was a high degree of correlation between adjacent memory locations. We also remark a similar bias between adjacent bytes for the *MSP430F2274* controllers.

Further investigations on the influence of environ-

mental factors, e.g., temperature or electromagnetic disturbances, on the state of SRAM are projected as future work for us.

## ACKNOWLEDGEMENT

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS-UEFISCDI, project number PN-II-RU-TE-2014-4-1501 (2015-2017).

## REFERENCES

Biham, E., Dunkelman, O., Indesteege, S., Keller, N., and Preneel, B. (2008). How to steal cars–a practical attack on keeloq. In *EUROCRYPT*, pages 1–18.

Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T., et al. (2011). Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*. San Francisco.

Francillon, A., Danev, B., Capkun, S., Capkun, S., and Capkun, S. (2011). Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS*.

Guajardo, J., Kumar, S. S., Schrijen, G.-J., and Tuyls, P. (2007a). *FPGA intrinsic PUFs and their use for IP protection*. Springer.

Guajardo, J., Kumar, S. S., Schrijen, G.-J., and Tuyls, P. (2007b). Physical unclonable functions and public-key crypto for fpga ip protection. In *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, pages 189–195. IEEE.

Holcomb, D. E., Burleson, W. P., and Fu, K. (2009). Power-up sram state as an identifying fingerprint and source of true random numbers. *Computers, IEEE Transactions on*, 58(9):1198–1210.

Holcomb, D. E., Burleson, W. P., Fu, K., et al. (2007). Initial sram state as a fingerprint and source of true random numbers for rfid tags. In *Proceedings of the Conference on RFID Security*, volume 7.

Ishtiaq Roufa, R. M., Mustafaa, H., Travis Taylora, S. O., Xua, W., Gruteserb, M., Trappeb, W., and Seskarb, I. (2010). Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *19th USENIX Security Symposium, Washington DC*, pages 11–13.

Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., et al. (2010). Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447–462. IEEE.

Krhovjak, J., Matyas, V., and Zizkovsky, J. (2009). Generating random and pseudorandom sequences in mobile devices. In *Security and Privacy in Mobile Information and Communication Systems*, pages 122–133. Springer.

Maes, R. and Verbauwhede, I. (2010). Physically unclonable functions: A study on the state of the art and future research directions. In *Towards Hardware-Intrinsic Security*, pages 3–37. Springer.

Miller, C. and Valasek, C. (2014). A survey of remote automotive attack surfaces. *Black Hat USA*.

Rührmair, U. and Holcomb, D. E. (2014). Pufs at a glance. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 347. European Design and Automation Association.

Shoukry, Y., Martin, P., Tabuada, P., and Srivastava, M. (2013). Non-invasive spoofing attacks for anti-lock braking systems. In *Cryptographic Hardware and Embedded Systems-CHES 2013*, pages 55–72. Springer.

Solomon, C. and Groza, B. (2015). Limon - lightweight authentication for tire pressure monitoring sensors. In *1st Workshop on the Security of Cyber-Physical Systems (affiliated to ESORICS 2015)*.

Studnia, I., Nicomette, V., Alata, E., Deswarte, Y., Kaâniche, M., and Laarouchi, Y. (2013). Survey on security threats and protection mechanisms in embedded automotive networks. In *Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on*, pages 1–12. IEEE.

Tillich, S. and Wójcik, M. (2012). Security analysis of an open car immobilizer protocol stack. In *Trusted Systems*, pages 83–94. Springer.

Toth, A. (2014). Method and system for monitoring a parameter of a tire of a vehicle. EP Patent App. EP20,120,464,019.

Verdult, R., Garcia, F. D., and Balasch, J. (2012). Gone in 360 seconds: Hijacking with hitag2. In *Proceedings of the 21st USENIX conference on Security symposium*, pages 37–37. USENIX Association.

Wetzels, J. (2014). Broken keys to the kingdom: Security and privacy aspects of rfid-based car keys. *arXiv preprint arXiv:1405.7424*.

Xu, M., Xu, W., Walker, J., and Moore, B. (2013). Lightweight secure communication protocols for in-vehicle sensor networks. In *Proceedings of the 2013 ACM workshop on Security, privacy & dependability for cyber vehicles*, pages 19–30. ACM.