CamPUF:

A CMOS-based PUF for device authentication

CfES Project - **Group 08**

Marco Chiarle s314730
Alessandro Vargiu s314294
Edoardo Manfredi s316714
Lorenzo Ruotolo s313207

Why CamPUF?

CamPUF is a new PUF design, based on commercial CMOS image sensors, which makes this design available for mobile devices.

It is designed for low-power devices and has a secure and light key generation mechanism. We are going to describe the theory concepts behind the design and then we will show the implementation.

Background

- PUFs
- Fixed Pattern Noise (PRNU, DSNU)
- Challenge-Response Authentication

PUFs

“**Physically Unclonable Function**” is used to address the challenges of hardware authentication, secure key generation, and anti-counterfeiting measures. PUF is a security metric that exploits inherent device physical variations to produce an unclonable and unique device response to a given input.

Why PUFs?

Pufs are stronger and more robust than other methods thanks to its principle characteristics as :

- Inherent Unclonability (Uniqueness and unpredictability)
- Resistance to invasive attacks
- No stored keys

Indeed if we consider a random number generator based on an algorithm with fixed seed, all of the characteristics said before are broken because of low entropy, predictability and vulnerability to attacks.

Examples of PUFs

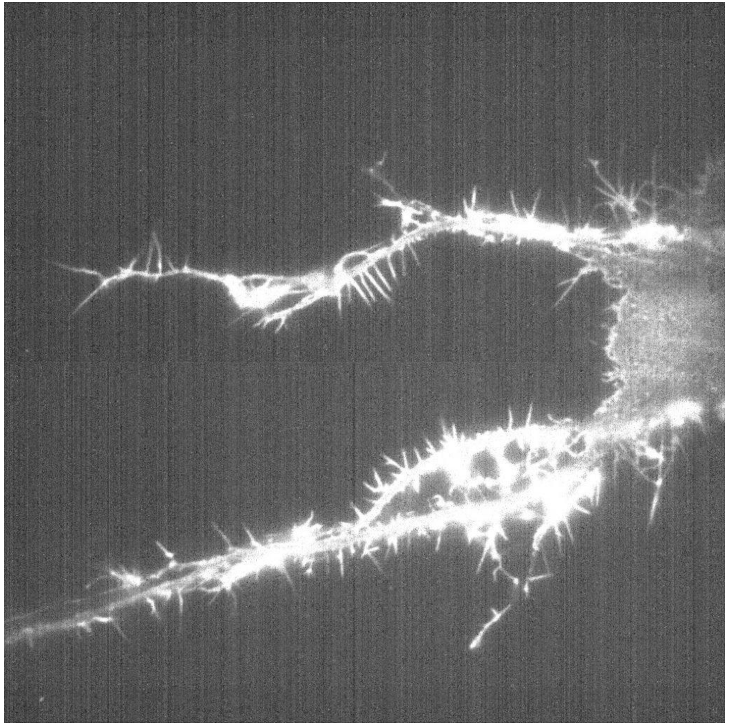
Different PUFs are available to use. They can be categorized based on main differences about randomness sources and measurement process .There are some examples:

- Delay PUFs
- SRAM PUFs
- CamPUFs

FPN (Fixed Pattern Noise)

A noise pattern present on imaging sensors (mainly CMOS-based ones), caused by differences in sensors pixels with respect to the average intensity.

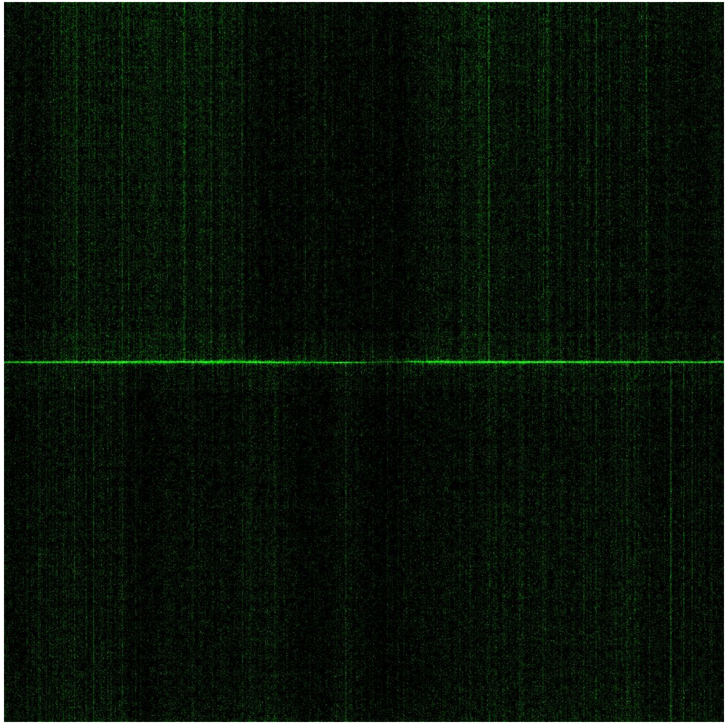
FPN is present in a particular position in space and has two sources: PRNU and DSNU



DSNU

Dark Signal Non-Uniformity occurs in dark images. To avoid negative values, each pixel has a positive offset called *bias*.

The fluctuation in the bias is the DSNU



Challenge-Response Authentication

Security protocol used to verify identity of a user by providing them a challenge to which the user must respond correctly.

Ex. OTPs, ZKPs, SSH-RSA...

CamPUF uses the DSNU Fingerprint of the device as a basis to the Challenge-Response mechanism.

CamPUF

- DSNU image sensor-based PUF
- Unique, light and stable key generation
- Dark image data processing guarantees protection, since most shared images are JPEG compressed or are illuminated.
- Minimal computation and light communication overhead

CamPUF:
Design and Implementation

Prerequisites

- device (**D**): untrusted entity that requires authentication.
- authenticator (**A**): trusted entity that authenticates **D** based on its registered challenge-response pair.
- challenge (**c**): sent by **A** to the device **D**.
- response key (**r**): key derived by **D**, in response of the challenge
- reference key (**r_ref**): key derived by **A**, and compared with **r** to authenticate **D**.

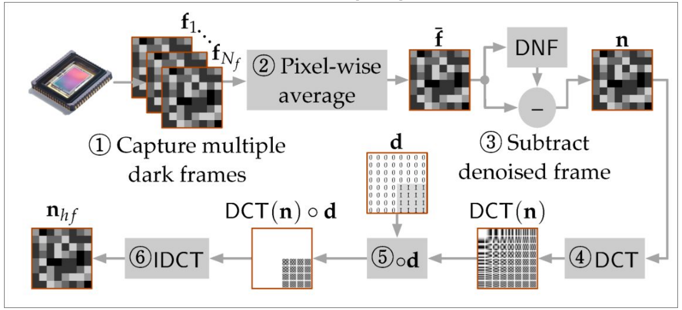
DSNU Fingerprint Extraction

What is it? DSNU fingerprint extraction is to obtain the unique noise pattern induced by the DSNU of **D**'s image sensor.

Where is it used? In CamPUF it is used in the authentication and enrollment phases.

Some properties:

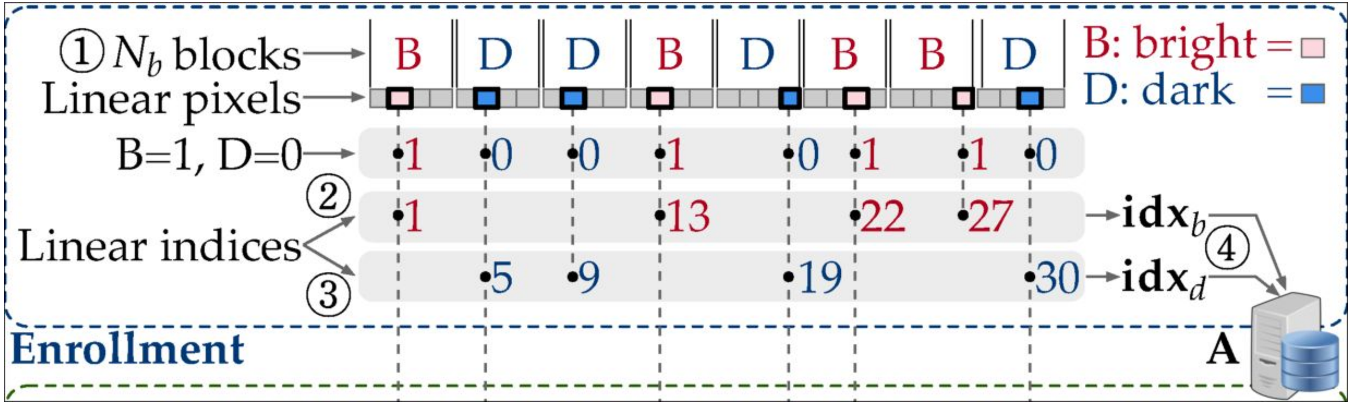
- It uses dark image(s) for enhancing the DSNU
- It should be as computationally light as possible, because it must be done locally by **D**.



Enrollment

The device **D** generates a short version of its DSNU fingerprint and registers it to the authenticator **A.**

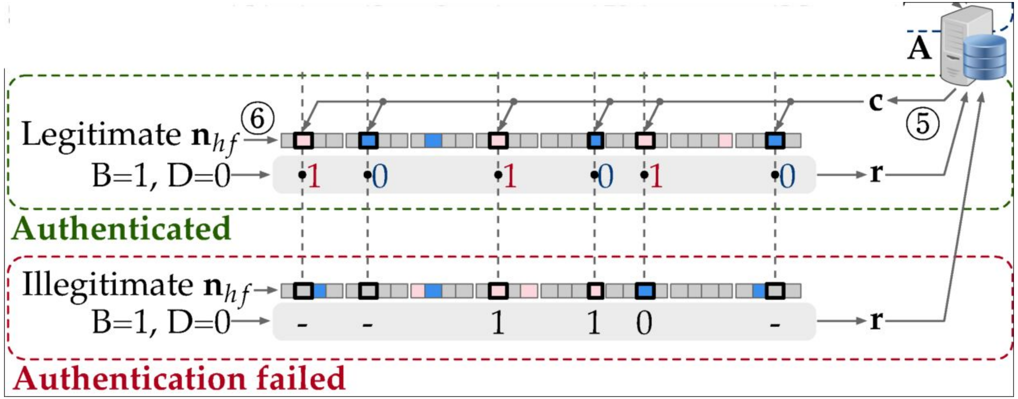
D's reference key **r_ref** is derived from the fingerprint, then the authenticator **A** does not need to store the key.



Authentication

The authenticator **A** sends a challenge created from the registered fingerprint and **D** generates a response key **r** based on the challenge and its DSNU fingerprint.

If **r** matches the reference key **r_ref**, **A** authenticates **D.**



CamPUF:
Testing & Results

The Dataset

Among multiple available options, this dataset was eventually chosen. It is composed of various sets of RAW images taken with five different IMX377 camera sensors, used by Android phones.

The photos are **completely dark** images, taken in different room temperatures: 25º, 35º and 45º.

The **absence of any light exposure** to the camera sensors is essential for an effective **DSNU extraction**. For real-world practical implementation, the images provided to the authentication algorithm should be taken in a similar way.

Testing the PUF

The testing was done using an automation script to try authentication on 50 different images after enrolling with a single image. The key length is 256.

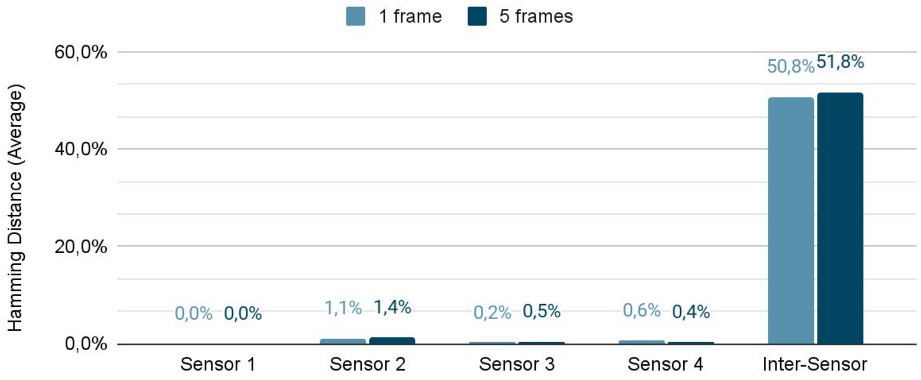
The expected Hamming Distance between the reference key **k_ref** and the response key **r** are:

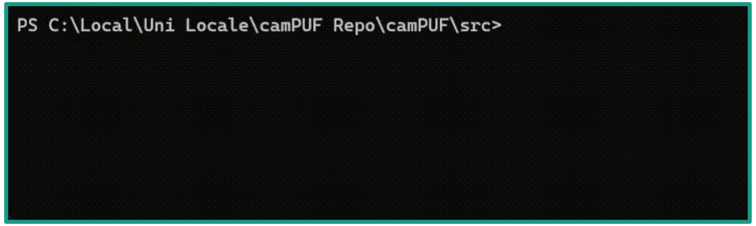
- Intra-Sensor Hamming Distance: ideally **zero**.
- Inter-Sensor Hamming Distance: ideally **high enough** to avoid false positives.

Results

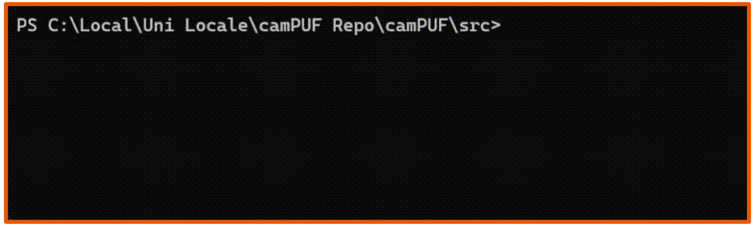
The averages obtained after testing:

- **Intra**-Sensor HD: <**2%**
- **Inter**-Sensor HD: >**50%**





Intra-Sensor

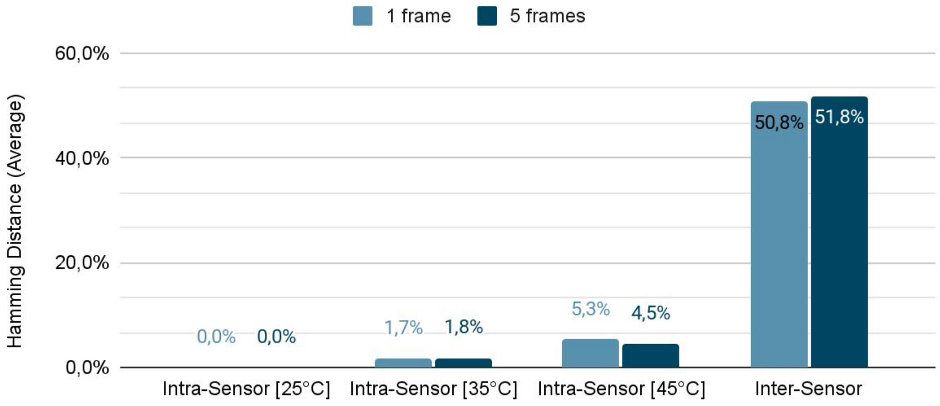


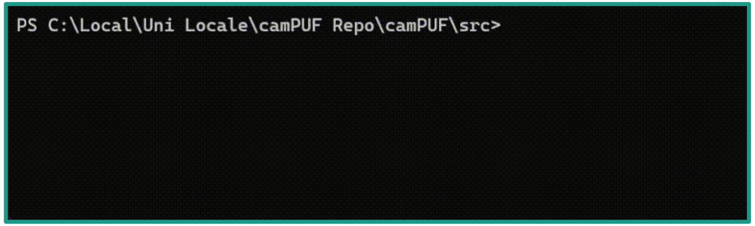
Inter-Sensor

Temperature

Images taken in **different temperatures** from the same sensor were also tested.

While the average Intra-Sensor HD rises with the temperatures, it is still **negligible** when compared with the Inter-Sensor HD.





Intra-Sensor [35°C]

Attacks Mitigation

RAW HF Noise

JPEG HF Noise

Main vulnerability: using **shared images** taken by the victim sensor.

Mitigation:

- Completely dark pictures are needed, uncommon to be shared.
- Even if obtained, **JPEG compression removes HF components**.

When trying to authenticate using dark JPEG images, the HD between keys is similar to Inter-Sensor values (>50%).

Summary

- CamPUF is a reliable PUF used to authenticate devices equipped with standard CMOS sensors.
- It exploits a quick **DSNU** extraction of easily obtainable dark pictures.
- The testings show the **uniqueness** of the key generated from each sensor.
- **Mitigation** of shared-images attacks granted by construction.
- CamPUF can be implemented with **no hardware modification** and a small software overhead.