# A Practical Guide to Differential Power Analysis of USIM cards

Christophe Devine, Manuel San Pedro et Adrian Thillard
christophe.devine@ssi.gouv.fr
mdsan.pedro@gmail.com
adrian.thillard@ssi.gouv.fr

ANSSI

**Abstract.** In 2015, Liu et al. [1] demonstrated an attack on USIM[1] cards that led to recovering authentication secrets using a differential power analysis attack. The compromise of those secrets allows an attacker to passively decrypt the subscriber's communications (such as texts), but also to impersonate the network and perform man-in-the-middle attacks. In this work, we wanted to assess if more recent cards were vulnerable, as well as establish whether the attack could be carried out with low cost, off-the-shelf equipment, and finally to provide recommendations.

## 1 Introduction

From a historical point of view, COMP128 was the first authentication algorithm to be used in GSM networks. Initially secret, an implementation was reverse-engineered and posted on the Internet in 1998 [2]. Several researchers then studied this algorithm and showed the possibility of recovering the shared master key (Ki) by capturing and analyzing between 50000 and 130000 challenges/responses from the SIM card. This led to the development of two new versions of COMP128 (v2 and v3), COMP128 v2 being identical to COMP128 v3 but with ten bits of the derived 64-bit key forced to 0. This new version of COMP128 remained secret until 2013 [3].

When 3G was being worked on in 1999, specifications were released to provide carriers with a new method for authentication: MILENAGE [4] [5]. Contrary to GSM, 3G networks require mutual authentication between the USIM card and the core network, and also require integrity protection; ciphering remains optional (as is the case in LTE), but integrity is mandatory. MILENAGE may also be used in a 2G context [6].

[1] Universal Subscriber Identity Module

In our attack scenario, we assume the attacker wishes to recover the secrets stored in the USIM card in order to decrypt the subscriber's communications at a later time. To do so, the attacker gains physical access to the card for a period of time. We will also assume the PIN code is trivial or has been disabled; this assumption is not too far fetched in France, as three operators amongst four use either 0000 or 1234 as the default PIN code.

### 1.1 MILENAGE

The MILENAGE algorithm is described in 3GPP Technical Specification 35.205 and 35.206 [4] [5]. Figure 1 presents a high-level view of this algorithm from the perspective of the USIM card:

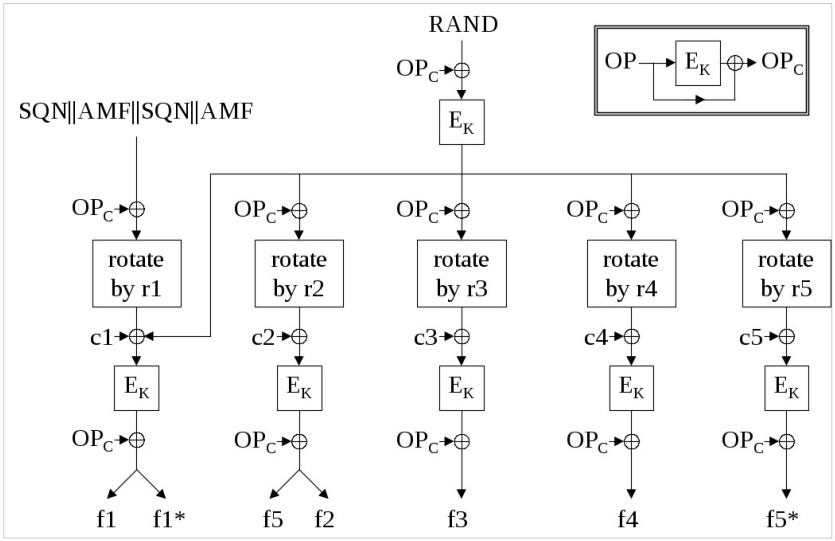


**Figure 1.** The MILENAGE algorithm (TS 35.206)

Here, $E_K$ is a 128-bit block cipher keyed with K. OPc is derived once from OP (a global constant set by the carrier) and stored in the USIM; there is no need to store OP itself. RAND and AMF are controlled by the network, and SQN is a shared counter that is incremented with each successful authentication, to protect against replay attacks. $r_1 ... r_5$ as well as $c_1 ... c_5$ are constants that can be set by the carrier (default values for these constants are found in the specification).

The output of MILENAGE are the keys used for ciphering and integrity of the radio traffic (f3 provides CK, f4 IK), and further values to authenticate both parties and resynchronize SQN if need be.

The attacker thus fully controls RAND and wishes to recover K, OPc and optionally $r_1...r_5$ / $c_1...c_5$. In theory, the carrier could choose any 128-bit block cipher, but since the specification provides an example based on AES, it seems likely certain carriers may choose AES by default.

### 1.2 Side-channel analysis

Side-channel attacks were introduced by Kocher et al in [7]. These attacks exploit the physical behavior of devices during the execution of an algorithm to recover information about the manipulated sensitive data.

Such attacks apply a classical *divide and conquer* strategy: the whole secret is divided into small parts (usually one byte), and each of these parts is attacked independently. To recover a secret value $k$, the attacker first identifies a simple *sensitive intermediate value* $v_k$ manipulated by the algorithm, which deterministically depends on both $k$ and a known input $p_i$.

Then, the attacker collects a large number of observations of the physical behavior of the device during the manipulation of $v_{k,i}$. We chose to observe the power consumption of the USIM card, which is a classical choice (another possibility being electromagnetic radiation). Each observation is related to the actual value of $v_{k,i}$ by a so-called *leakage function* $\ell(\cdot)$. The attacker hence gets a vector $\ell(v_{k,i})_i$ characterizing the manipulation of the data while being provided a constant key $k$ and varying inputs $(p_i)_i$.

The attacker then exhausts all possible values for $k$. For each hypothesis $\hat{k}$ and for each input $p_i$, he computes the corresponding sensitive variable $(v_{\hat{k},i})$ hypothetically manipulated by the device. Then, the attacker chooses a leakage function $m$ to map the value of the sensitive data towards the estimated leakage. He hence obtains $(h_{\hat{k},i})_i$, where for every $i$, we have $h_{\hat{k},i} = m(v_{\hat{k},i})$. Here, we followed [1] and chose to model the leakage as the Hamming weight function, denoted $HW$.

Finally, the attacker compares the leakages $(\ell_{k,i})_i$ obtained in the second phase with all the hypotheses $(HW(v_{\hat{k},i}))_i$ he constructed. This comparison is done using a *statistical distinguisher* and lays the most likely value for the used key; here, we use the Pearson linear correlation coefficient as a statistical distinguisher.

### 1.3 Attacking MILENAGE

Based on the previous theoretical model, the MILENAGE algorithm can be attacked in the following way: first, we know RAND as it is an attacker-controlled input. RAND is XORed with OPc, and then XORed with the first round key (identical with K) to be input to the AES S-box in the SubBytes stage. The output of the S-box will be the target of the first pass of the attack.

**Algorithm 1** Deriving the prediction matrix

**for** each of the 16 bytes of ($OPc \oplus K$) under attack (i=0..15) **do**
  **for** each MILENAGE call associated with an attacker chosen RAND **do**
    **for** each of the 256 hypothesis about ($OPc \oplus K$) **do**
      XOR the hypothesis with RAND[i],
      Process the previous value through the AES S-box,
      Compute $HW$ of the previous value: this is the prediction.

As a simple example, when attacking the first byte of $OPc \oplus K$, if the hypothesis is 0x11 and RAND[0] was chosen as 0x9A, the result will be $HW(Sbox[11 \oplus 9A]) = HW(Sbox[8B]) = HW(3D) = 5$.

This algorithm provides us, for each byte under attack, with a prediction matrix composed of 256 rows and $n$ columns (the total number of calls to MILENAGE that were made). In addition to the prediction matrix, the power consumed by the USIM card is measured with each MILENAGE call. Let's assume $m$ is the number of points in a power trace: then we have a second matrix composed of $n$ rows and $m$ columns. Through the correlation of the two matrices, one correct hypothesis should emerge for each byte (i).

If the first stage was successful in recovering $OPc \oplus K$, we can then deduce (after ShiftRows and MixColumns) the input that will be XORed with the second stage round keys RK2. Attacking again the output of the

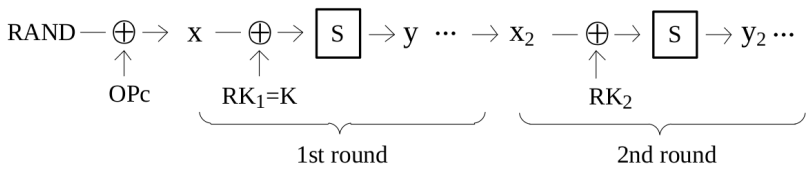


**Figure 2.** First two AES rounds (from [1])

S-box, RK2 can be recovered leading to K itself.

Constants $r_1$ through $r_5$ and $c_1$ through $c_5$ can be recovered as well; we refer the reader to the aforementioned article [1] which details the complete procedure.

## 2 Test setup

### 2.1 Card reader

The first building block in our setup is the ability to craft commands to be sent to the USIM card and call MILENAGE. To this end, we used the *card* library [8] to derive the correct APDUs, as illustrated in the following code snippet (for a more complete understanding of the commands being sent, we refer the reader to section 7.1.2 *Command parameters and data* of TS 31.102 [9]).

```
selectFile1 = [0x00, 0xA4, 0x08, 0x04, 0x02, 0x2F, 0x00]
headerReadRecord = [0x00, 0xB2, 0x01, 0x04]
headerSelectFile2 = [0x00, 0xA4, 0x04, 0x04]
headerGetResponse = [0x00, 0xC0, 0x00, 0x00]
InternalAuthenticate = [0x00, 0x88, 0x00, 0x81, 0x22]

def milenage_init(reader):
    answer,sw1,sw2 = reader.send_apdu(selectFile1)
    answer,sw1,sw2 = reader.send_apdu(headerGetResponse + [sw2])
    answer,sw1,sw2 = reader.send_apdu(headerReadRecord  + [answer[7]])
    answer,sw1,sw2 = reader.send_apdu(headerSelectFile2 +
                     list(answer[3:4 + answer[3]]))
    answer,sw1,sw2 = reader.send_apdu(headerGetResponse + [sw2])

def milenage_request(reader, rand = [0]*16, autn = [0]*16):
    return reader.send_apdu(InternalAuthenticate + [16] +
                            list(rand) + [16] + list(autn), True)
```

In addition, a custom card reader was used. It is composed of a 8-bit microcontroller with its serial port connected to the PC through USB.

It controls the clock and I/O lines to the smartcard under test, and additionally provides a trigger output to notify the oscilloscope to start an acquisition. Finally, a 10 Ohm resistor is inserted after the GND pin of the smartcard; of which voltage is mesured across, providing how much power is being consumed.

While the software and hardware parts of this card reader are not open-source, we would like to mention the WooKey project [10], to be released in 2018. This open-source and open-hardware project may be used as a card reader (it includes an ISO 7816 implementation), and can be modified to add a trigger signal after sending an APDU such as INTERNAL AUTHENTICATE.

Another platform for side-channel attacks worth mentioning is the ChipWhisperer; it provides a complete open-source software and hardware toolchain for performing side-channel power analysis of various targets [11].

### 2.2 Oscilloscope setup

Initially, we operated a high-end Lecroy oscilloscope capable of up to 10 Gsample/second and 1 GHz bandwidth for the characterization of the USIM cards being tested. After a successful attack on one of the cards, we reproduced it on an entry-level Rigol oscilloscope, the DS1054Z. It costs about 400€ and can acquire at up to 1 GS/s with 100 MHz bandwidth.

The DS1054Z project developed by Philipp Klaus [12] was used to setup the oscilloscope in SINGLE mode, and acquire raw waveforms over Ethernet with the `_get_waveform_bytes_internal` function. In order to accelerate this acquisition phase, `_get_waveform_bytes_internal` was altered to remove setup calls that were only necessary once.

During acquisition, the issue of electromagnetic noise became apparent. One PC in particular would leak large amounts of noise through the ground and shield of the USB cable. The proper remedy would be in that case to provide a form of electrical isolation with an optocoupler. We found a temporary workaround: by cutting all lines in the USB cable, except D+ and D-, communications with the microcontroller still remained but the noise coming from the PC disappeared.

Also, nearby devices transmitting EM radiation would couple on the setup and insert noise into the measurements. Figure 3 shows the emission of a nearby phone, in this case transmitting a GSM burst. Ideally the measurement should be run within a Faraday cage, but another solution is to power off nearby devices, or put them as far away as possible.

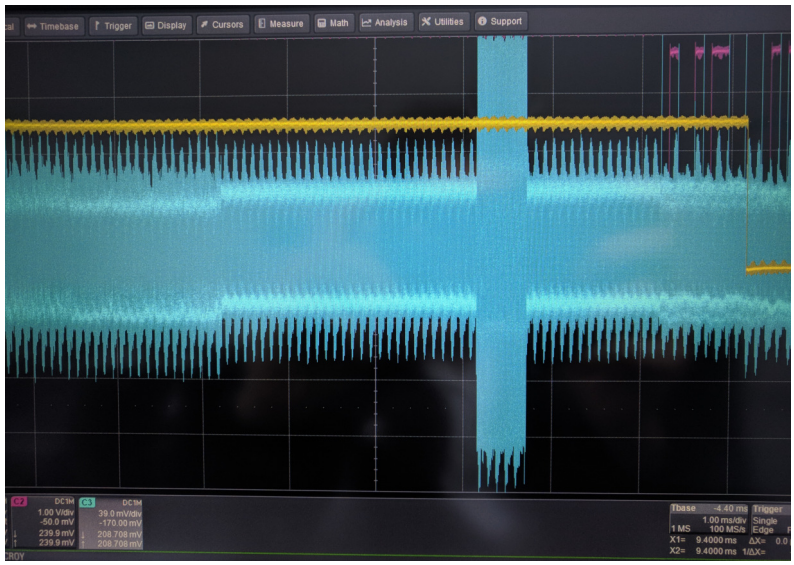


**Figure 3.** GSM burst leaking into the test setup

To sum it up, here is an outline of the script that performs acquisition of traces. `milenage_request` will lead the trigger signal to be held high until a response is received.

**Algorithm 2** Acquiring power consumption traces

Connect to the oscilloscope
Setup the USIM with `milenage_init`
Create the HDF5 output file
**for** each trace to be captured **do**
    Set the oscilloscope in SINGLE mode,
    Pick a random RAND 16-byte value,
    Call `milenage_request` with RAND,
    Get waveform bytes with `_get_waveform_bytes_internal`,
    Store RAND and the waveform in the HDF5 output file.

## 3 Results

We tested nine USIM cards (amongst which five are from french carriers); one non-French card was found to be vulnerable to the attack. This is not to say that other cards are not vulnerable at all, but rather that more work might be needed to achieve success.

In the following, we will concentrate on the vulnerable USIM card[2].

### 3.1 Identification of MILENAGE

Figure 4 shows the result of one call to MILENAGE. Channel 1 (yellow) is the trigger, channel 2 (purple) is the I/O line to the USIM and channel 3 is the power consumption. We can see the processing of this APDU takes about 52ms.


**Figure 4.** Power consumption of the vulnerable card

While at this stage we cannot distinguish MILENAGE, zooming (figure 5) shows a blocks repeated six times, possibly corresponding to the six call to $E_k$ as shown in figure 1.

Further zooming (figures 6, 7) on one of the blocks shows a pattern of ten peaks leading to further suspicion those are the ten rounds of AES-128, which would be $E_k$.

[2] The maker of this card was notified of our findings in January 2018, and is working on a fix.

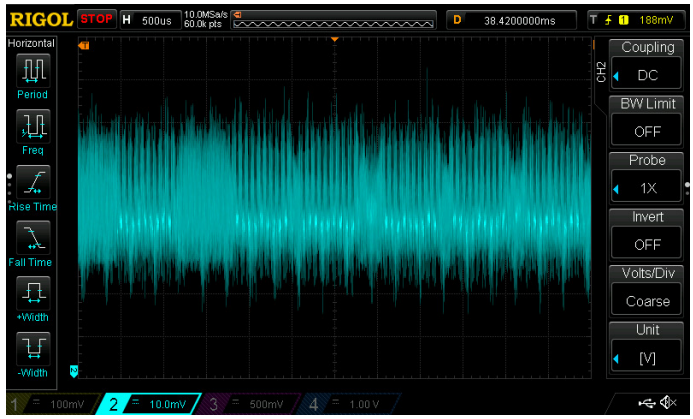


**Figure 5.** Zooming on the MILENAGE algorithm

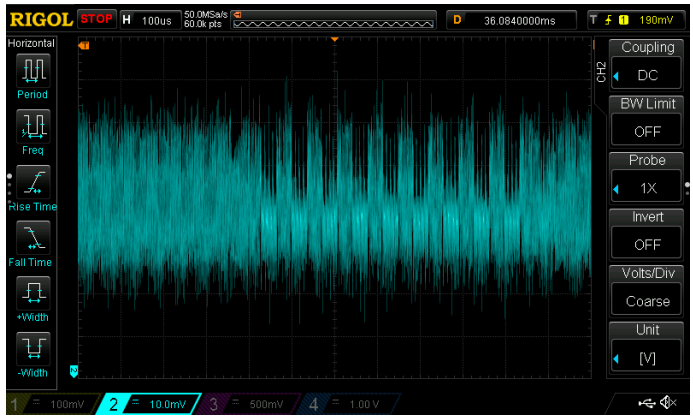


**Figure 6.** Zooming on one call to $E_k$ (AES) within MILENAGE

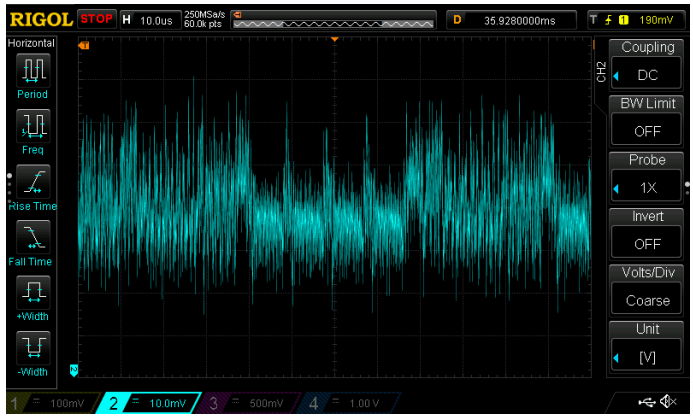


**Figure 7.** Zooming on the first round of the first call to AES

While the MILENAGE implementation in this particular USIM card could be located by eye only, this is not always the case, especially in the case of countermeasures that smooth out power consumption. Statistical tools exist to determine at which point the data we control (here, RAND) is being manipulated; NICV[3] is a popular tool for this purpose [13].

### 3.2 Synchronizing the traces

Before conducting the Correlation Power Analysis itself, it is required to temporally synchronize the traces. As the clock is not perfect, jitter may happen leading to traces not starting exactly at the same point in time.

In the following example, we captured 4000 traces (having 60000 points each) of the beginning of the first AES call, at a sampling rate of 125 MS/s. In figure 8 are shown two traces taken from the whole set of traces; a clear temporal delay between the two traces is present.

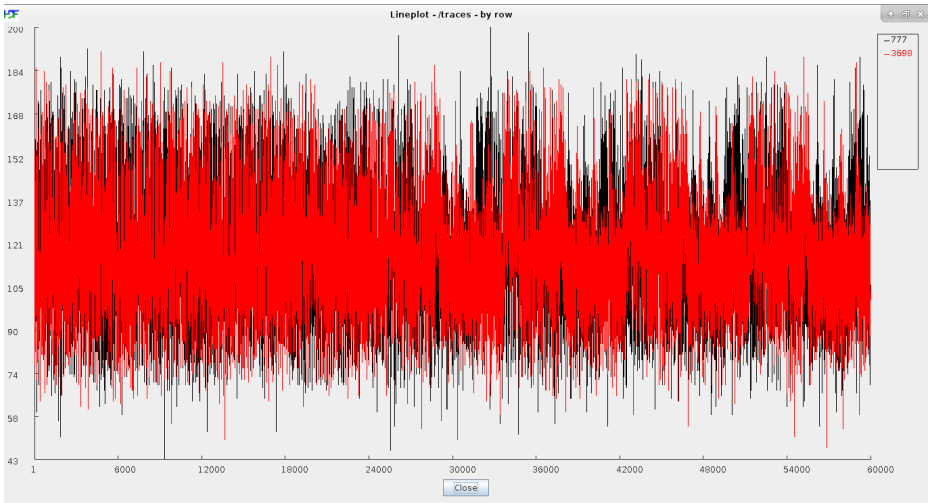


**Figure 8.** Two power consumption traces, before synchronization

To synchronize the traces, a Python script takes the first trace as a reference and tries to shift each trace until a correlation maximum is found (using `numpy.corrcoef`), then performs a rotation of the trace; figure 9 shows the same two traces after synchronization.

[3] normalized inter-class variance

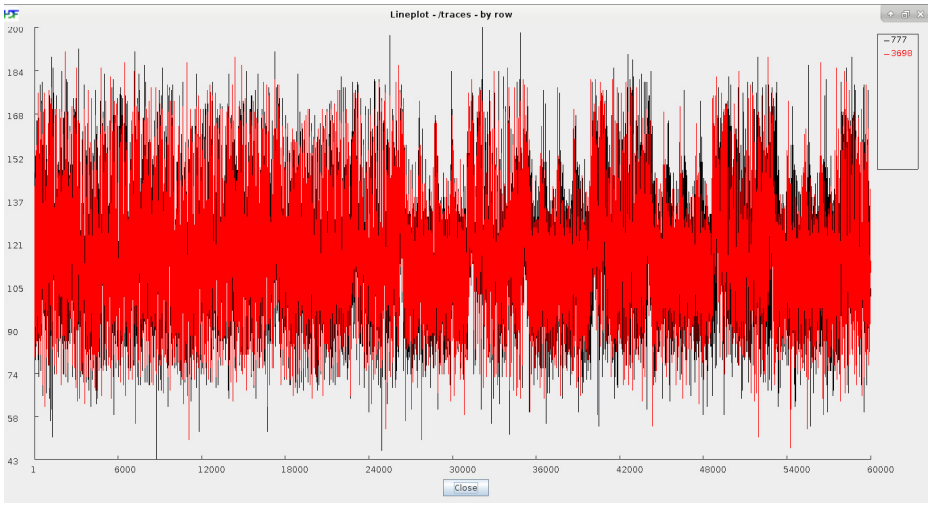


**Figure 9.** Two power consumption traces, after synchronization

### 3.3 Correlation Power Analysis

The final step in this attack is to recover each secret: $OPc \oplus K$, then RK2, one byte at a time, by correlating the prediction matrix with the captured traces (using for instance `numpy`). In our case, predictions are contained within a 256x4000 matrix. By correlating this matrix with the 4000x60000 matrix containing traces of power consumption, we obtain a 256x60000 matrix where we can look for the hypothesis leading to the highest correlation value, for a given key byte.

Figure 10 shows, for the 8th byte of $OPc \oplus K$, a selection of three hypothesis. Hypothesis 168 is the correct one and, amongst all hypothesis, presents several high peaks where the sensitive variable is being manipulated; we thus deduce that $OPc \oplus K[7] = 168$.

About 1.2 traces/second can be captured on the RIGOL DS1054Z; 4000 traces are acquired in 80 minutes. After this acquisition phase, resynchronisation and correlation takes ~20mn on a quad-core PC.

The final step, after having recovered OPc and K, is to verify if they are correct. This is done by simply calling MILENAGE and validating the MAC (SQN resync may be performed as well). Success indicates the constants $r_1 ... r_5$ and $c_1 ... c_5$ were not changed by the carrier and do not need to be recovered.

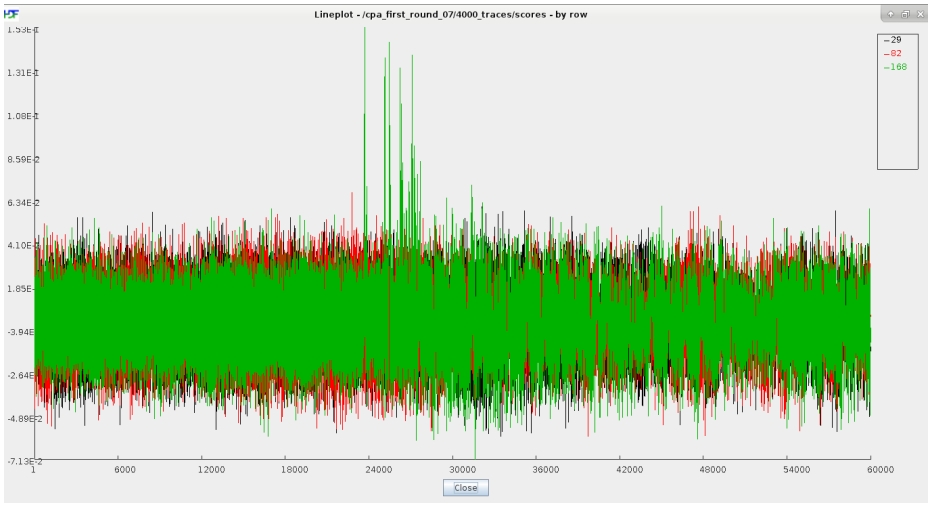


**Figure 10.** Three hypothesis are shown here for the 8th byte of $OPc \oplus K$

## 4 Conclusion

Our work has shown that the attack presented in [1] could be reproduced even using low-cost measurement equipment and open-source components. Although most USIM cards were not successfully compromised, we should not assume they are invulnerable; dedicating more time and pushing the attack (for instance, using SCARE[4] [14] [15] or a higher order attack [16]) could lead to success.

From the point of view of the user, we recommend changing the PIN (and not disable PIN security), especially if the carrier's default code is trivial. While PIN verification could itself be the suject of a phyical attack, it remains nonetheless a deterrent that will slow down the attacker.

Countermeasures exist to protect against side-channel analysis. Certified secure ICs are available, and we encourage carriers to choose such ICs. Finally, it is our hope that future generation networks will move away from authentication based on symmetric-key algorithms, and instead will embrace more modern methods that would enable desirable security features such as Perfect Forward Secrecy.

[4] Side channel analysis for reverse engineering

## Acknowledgement

We would like to thank all our present and past collegues who have helped us with this work (in no particular order): Benoit Michau, Victor Lomné, Marc Blanc-Patin and Ryad Benadjila.

## References

1. J. Liu, Y. Yu, F.-X. Standaert, Z. Guo, D. Gu, W. Sun, Y. Ge, and X. Xie, "Small Tweaks do Not Help: Differential Power Analysis of MILENAGE Implementations in 3G/4G USIM Cards," *Black Hat USA Briefings*, 2015.
2. M. Briceno, I. Goldberg, and D. Wagner, "An implementation of the GSM A3A8 algorithm. (Specifically, COMP128.)," 1998. `http://www.iol.ie/~kooltek/a3a8.txt`.
3. `https://doc.freeradius.org/comp128_8c_source.html`.
4. "3GPP TS 35.205 - MILENAGE algorithm set: General." `http://www.3gpp.org/DynaReport/35205.htm`.
5. "3GPP TS 35.206 - MILENAGE algorithm set: Algorithm specification." `http://www.3gpp.org/DynaReport/35206.htm`.
6. "3GPP TS 55.205 - GSM-MILENAGE algorithms: Functions A3 and A8." `http://www.3gpp.org/DynaReport/55205.htm`.
7. P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Annual International Cryptology Conference*, pp. 104–113, 1996.
8. `https://github.com/mitshell/card`.
9. "3GPP TS 31.102 - Characteristics of the USIM application." `http://www.3gpp.org/DynaReport/31102.htm`.
10. R. Benadjila, J. Lefaure, A. Michelizza, M. Renard, P. Thierry, and P. Trebuchet, "WooKey: USB Devices Strike Back," *Symposium sur la sécurité des technologies de l'information et des communications*, 2018. `https://github.com/wookey-project/`.
11. `https://newae.com/tools/chipwhisperer/`.
12. `https://github.com/pklaus/ds1054z`.
13. S. Bhasin, J.-L. Danger, S. Guilley, and Z. Najm, "NICV: normalized inter-class variance for detection of side-channel leakage," in *Electromagnetic Compatibility, Tokyo (EMC'14/Tokyo), 2014 International Symposium on*, pp. 310–313, 2014.
14. C. Clavier, "Side channel analysis for reverse engineering (SCARE) - an improved attack against a secret A3/A8 GSM algorithm," *International Conference on Information Systems Security*, 2004.
15. C. Clavier, "An Introduction to Physical Attacks - Application to Secret Specifications Algorithms," *Symposium sur la sécurité des technologies de l'information et des communications*, 2007.
16. E. Prouff, M. Rivain, and R. Bevan, "Statistical analysis of second order differential power analysis," *IEEE Transactions on computers*, vol. 58, no. 6, pp. 799–811, 2009.