*cryptography*

MDPI

*Article*

# Locating Side Channel Leakage in Time through Matched Filters

**Alessandro Barenghi *, Gioele Falcetti and Gerardo Pelosi ***

Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, 32, 20133 Milan, Italy; gioele.falcetti@mail.polimi.it

* Correspondence: alessandro.barenghi@polimi.it (A.B.); gerardo.pelosi@polimi.it (G.P.); Tel.: +39-02-2399-3476 (A.B.)

**Abstract:** Side channel attacks provide an effective way to extract secret information from the execution of cryptographic algorithms run on a variety of computing devices. One of the crucial steps for a side channel attack to succeed is the capability to locate the time instant in which the cryptographic primitive being attacked is effectively leaking information on the side channel itself, and synchronize the data obtained from the measurements on that instant. In this work, we propose an efficient and effective solution relying on the digital signal processing technique known as matched filters. We derive our matched filter with a small amount of profiling information which can be obtained from a device matching the one under attack. Our technique reliably identifies the cryptographic operation being computed, even when system interrupts or software multithreading are enabled on our target platform. We validate our approach through a successful attack against an unprotected AES implementation running on a Cortex-M4-based microcontroller.

**Keywords:** side channel attacks; electromagnetic emission attacks; localization of cipher computation; computer security



**Citation:** Barenghi, A.; Falcetti, G.; Pelosi, G. Locating Side Channel Leakage in Time through Matched Filters. *Cryptography* **2022**, *6*, 26. https://doi.org/10.3390/cryptography6020026

Academic Editor: Josef Pieprzyk

Received: 23 April 2022
Accepted: 30 May 2022
Published: 30 May 2022



## 1. Introduction

Side channel attacks have proven to be a concrete threat to the ubiquitous computation devices embedded in common objects, which have an increasing requirement of security features [1]. Indeed, low end computational platforms such as microcontrollers and simple Systems on Chips (SoCs) are likely to be deployed in a non-controlled environment where an attacker is able to achieve physical proximity to them. We refer the reader interested to a survey on the current techniques to the following work [2].

One of the necessary assumptions to be made for a side channel attack is that the adversary should be able to locate when the security sensitive operations take place during the computations executed on the device at hand, e.g., the ones corresponding to the encryption transformation performed via a symmetric-key cipher. In particular, some sort of step-lock synchronization between the sensitive computation and a measurable signal, which is employed as a start-of- measurement trigger, is commonly used in attacking real world computation platforms. While this approach was proven to be practical on a variety of devices ranging from hardware security modules [3] to microcontrollers [4] and large SoCs [5,6], it remains a challenging task to automate the detection of the informative portion of a side channel signal (without an explicit trigger), and automate either its time-aligned acquisition, or post-acquisition alignment and segmentation.

The difficulty of individuating the relevant portion of a side channel measurement, namely, either a power consumption or Electro-Magnetic (EM) emissions *trace*, has inspired a countermeasure for microcontrollers reported in [7]. In the aforementioned work, the authors propose to employ time-sharing multithreading on a single-core microcontroller to hinder the attacker's ability to perform useful (i.e., time aligned) side channel measurements. This comes at the cost of a timing overhead due to the context-change

among threads, and the interleaved execution of computations which only act as a noise-generation process. On the front of cryptographic primitive identification from side channel leakage, the authors of [8] proposed a method to select which cryptographic primitive, among a set of profiled ones, is being executed on an ESP32 embedded microcontroller platform. The approach adopted in [8] also performs trace realignment after classification; in particular, it compares three different classification strategies (mean-square error minimization in both time and frequency domain and neural network classification), achieving a 96.47 classification accuracy and a 34 µs timing jitter in realigning the primitive executions. Our approach compares favourably to the one proposed in [8], in terms of computational efficiency of deriving the template (we only require averaging a set of collected values to derive the template) and in terms of data required to build the template itself (we achieve a classification accuracy above 99.9% with a single trace as a template, and an alignment jitter below 1 µs). We note, however, that our environment is less affected by strong disturbances, as the case study platform of [8] is endowed with a Wi-Fi radio front-end, which is enabled during side channel measurements. A recent work [9], tackles an attacker model with a further relaxation in the requirements, i.e., the attacker only knows some technical details of the target platform (such as the operating frequency) and the structure of the cryptographic primitive at hand (e.g., it is constituted by a countable loop, as it is a symmetric block cipher). The authors of [9] show how it is possible to turn this knowledge into an operational approach to semi-automate the task of locating the moment in which the cryptographic primitive is being executed. With respect to this work, our approach involves a more restrictive attacker model (as we require another instance of the target device to be used as template device), but, differently from others, it is fully automated.

**Contribution.** We propose a computationally efficient technique relying on *matched filters* to identify the position in time when a software implementation of the Advanced Encryption Standard (AES) [10] cipher is being run. Our approach follows the established praxis in communications which identifies the presence of a modulated symbol on the channel starting from the knowledge of the ideal form of the modulated symbol itself. Our proposed identification and segmentation technique is able to effectively overcome the side channel countermeasure reported in [7] in all cases where the number of context-changes is lower than the one of the executions of the cryptographic primitive itself, and the workload being computed by the hiding threads differs from an AES computation. We note that such a scenario is expected to be common when symmetric ciphers are at hand, as interrupting the cryptographic computation process more than once per cipher execution would imply a significant computational overhead.

The remainder of this work is organized as follows. Section 2 provides background information on side channel attacks and digital signal processing, and describes our matching method and its rationale. Section 3 reports the results of our experimental validation campaign, carried out on a commercial grade microcontroller (without embedded side channel attack countermeasures). Finally, Section 4 reports our conclusions and highlights directions for future works.

## 2. A Matched Filter Approach to Cryptographic Primitive Detection

The majority of side channel attacks exploiting the power consumption of a device or its EM radiated emissions rely on the ability of collecting multiple samples of the side channel value corresponding to a given elementary computation (e.g., a single Boolean operation). The need for multiple samples arises from the natural presence of noise, which affects the measured value of the side channel. To the end of coping with the noise, the side channel measurement is thus considered as a sample from a random variable. Since the noise is assumed to be Gaussian, with zero mean, and additive, side channel attack techniques cope with it by averaging out its effects over multiple samples. Our task in this work is to show that an adversary is able to perform a side channel measurement of multiple computations of the same cryptographic primitive and segment it along the time axis. The segmentation produces a set of time sequences of side channel measures

known as a set of *traces*, which are *aligned*, i.e., for any time instant, all the traces contain a side channel measurement pertaining to the same operation. To this end, we rephrase the problem of the detection of the computations of a cryptographic primitive starting from a side channel measurement of the device computing it, as the one of detecting a known signal in the side channel measurement.

A *matched filter* is a Digital Signal Processing technique which was proposed to detect the presence of a known signal, denoted as *template*, within an arbitrary unknown signal [11]. In particular, the matched filter maximizes the Signal-to-Noise (SNR) ratio in case the unknown signal contains the template affected by additive stochastic noise. The output of the matched filter can be interpreted as a measure of the correlation between the template signal and the input signal in any given instant of time. As a consequence, high values being output by a matched filter indicate a high likelihood of the template being present in the analyzed signal at the corresponding time instant. In case multiple repetitions of the template are present in the analyzed signal, the matched filter will output multiple local maxima indicating the position of the replicas. Considering the case of real, discrete-time signals, we denote as $t(n)$ the template signal. In our concrete case, the template signal is non null only for a finite amount of time instants $N > 0$, starting from 0. The impulse response of a causal matched filter $h(n)$ is defined as $h(n) = t(N - n)$. The reason for the definition of a causal matched filter is to allow an online application of our method, should the need arise. Indeed, while our scenario foresees the digital signal processing phase taking place after the entire measurement of the side channel took place, it can be useful to perform online detection of a cryptographic primitive. As a consequence, the output of the matched filter $y(n)$ is defined as a function of its input signal $x(n)$ as:

$$y(n) = \sum_{k=-\infty}^{\infty} h(n-k)x(k) = \sum_{k=-\infty}^{\infty} t(N+k-n)x(k)$$

It is useful to recall that, as we defined a causal matched filter, the amplitude of its output $y(N)$ at time $N$ quantifies how much the unknown signal $x(n)$ in the interval $[0, N-1]$ is matching the template. This is in contrast with an anti-causal matched filter which outputs the correlation of the unknown signal in the $[0, N-1]$ interval as $y(0)$.

The key intuition underlying our proposal to use a matched filter to detect the presence and position in time of a cryptographic primitive is that a dominant portion of the side channel signal is determined by the type of the instruction being executed, while the involved data provide a variation of a lesser extent. Indeed, considering the case of software implementations of cryptographic primitives, a significant portion of the computing hardware (i.e., the CPU) is dedicated to the management of the instruction flow (i.e., fetching and decoding instructions). As a consequence, the portion of the side channel leakage behaviour induced by it will contain information on which instructions are being executed. We note that, in the case of an ASIC implementation of a cryptographic primitive, such a portion of the energy may be smaller, but will still be present, as the control circuitry (e.g., the one implementing the driving Finite State Machine), must be present. A concrete substantiation of our intuition is that Simple Power Analysis (SPA) [12] techniques, which rely on portions of code being conditionally executed depending on a secret value, observe differences in the side channel behaviour, caused by different instructions being executed, relevant enough to be told apart by a human observer.

We thus consider, as our template-signal to be found, the side channel behaviour of the device computing the cryptographic primitive of the attacker's choice. We clarify that the template behaviour to be derived is not data dependent, as it is the case in for the homonymous side channel attack strategy known as *Template Attacks* [13]. In order to obtain the template, we consider that the attacker has access to a device of the same model as the one in which the cryptographic operation is being executed, and on which she has full control of the execution. Considering the case of software cryptographic implementations, this is equivalent to obtaining a copy of the same microcontroller/SoC being targeted,

and being able to run the same cryptographic implementation on it, while recording its side channel behaviour. We note that the attacker does not need control over the data being processed in the device, relaxing the (accepted) attacker model of [13].

Once a template is available to the attacker, she will proceed to measure the side channel behaviour of the device under attack, where the cryptographic primitive computations need to be located in time, and apply the matched filter. Employing the output of the filter itself will allow her to segment the trace. We note that, in case both the signal and the template are normalized, the output of the matched filter on a perfect match will be 1. However, since normalization may be challenging in our context, we apply the following approach: we consider any output of the matched filter above a threshold, chosen as a fraction of the maximum observed output, to be the result of a template match. This strategy can be automatically tuned after the detection of at least one instance of the template in the unknown signal. As a consequence, its adaptation to the online detection case requires the attacker to perform two runs of the computation to be attacked on the target device: the first run will provide the means to detect the maximum value output by the matched filter, while the second run will be employed to detect and segment the actual cryptographic primitive traces.

## 3. Experimental Evaluation

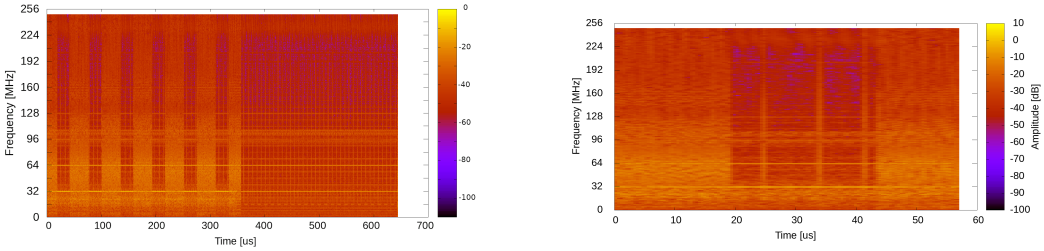
We selected as a target device for our experimental validation a ST-NUCLEO-F401RE [14], board developed by STMicroelectronics. The board is equipped with an ARM Cortex-M4 based STM32-F401RE microcontroller. We ran the STM32-F401RE microcontroller at 32 MHz during our experiments. We took as our case study the Advanced Encryption Standard (AES) [10] cipher with 128-bit key. We considered the official implementation of AES-128 currently employed in the mbedTLS [15] software library, which relies on the use of T-Tables to perform the computations prescribed for each of the 10 iterated rounds composing the cipher. The implementation is currently available at https://github.com/Mbed-TLS/mbedtls/blob/development/library/aes.c (accessed on 20 April 2022).

To obtain time-sharing multithreading on the board, we performed all our measurements while running the Miosix [16] Operating System (OS) for 32-bit embedded devices. In our measurements, Miosix provided both the multithreading capabilities and acted as a hardware abstraction layer.

Our measurement setup was constituted by a PicoScope 5244D [17], digital sampling oscilloscope, quantizing the $\pm$500 mV range on 8 bits, and sampling at a frequency of 500 M samples per second. We employed two Agilent INA-10386 amplifiers, with a gain of 26 dB each, in a cascade configuration, obtaining a total of 52 dB of gain and a simple custom loop probe obtained out of a single turn of 50 $\Omega$ coaxial cable which was placed on top of the STM32-F401RE package. All the computations were carried out on a desktop with an AMD A8-6600K CPU and 16 GiB of DDR3 DRAM, running Debian GNU/Linux 10. The template generation, identification and segmentation software was realized as a standalone C++ program compiled with g++ 8.3.0.

**Validation of the dominating role of instruction mix on side channel features.** Our first step in our experimental validation was to assess the presence of distinctive features of the AES computation with respect to the idle behaviour of our target platform. To this end, we sampled the EM emissions for six AES computations, followed by a loop iterating nop instructions. Figure 1 depicts the amplitudes of the Short-Time Fourier Transform (STFT) applied to the sampled signal with a frequency resolution of 2 MHz. From Figure 1a it is possible to distinguish clearly the periodic behaviour of the relevant harmonic components of the signal due to the iterated computation of the AES algorithm from the iterated computation of the NOP operations. Furthermore, analyzing the time interval corresponding to a single AES computation (Figure 1b), it is possible to distinguish three regions, roughly 20 µs in length, according to the harmonic components having different magnitudes. It also noteworthy that the backward jumps constituting the backedge of the loops of both the AES rounds and the loop iterating nops share the same high magnitude of

most spectral components (they appear as light orange vertical lines). These observations confirm that a significant portion of the spectral features of the EM side channel depend on the instruction mix being executed by the microcontroller.




(**a**) (**b**)

**Figure 1.** Output of the Short Time Fourier transform computed on a sequence of six AES computations followed by a loop iterating nop instructions. (**a**) Six AES computations followed by a loop iterating nop instructions. (**b**) Close-up of a single AES computation.

**Identification of timing location of AES.** Our first validation test aims at assessing the effectiveness of a matched filter, given the amount and quality of information required to build its template-signal. To this end, we built four different templates as follows: (i) a single sampled trace from an AES computation with a randomly drawn plaintext and key; (ii) the time-wise average of 1000 traces of AES computations with the same (randomly drawn) plaintext-key pair; (iii) the time-wise average of 1000 traces of AES computations with randomly drawn plaintexts and a fixed (randomly drawn) key; and (iv) the time-wise average of 1000 traces of AES computations with randomly drawn plaintexts and keys. The purpose of the four templates is to mimic the following scenarios: (i) the attacker singles out by hand one cipher computation from either an existing platform or some position in the available side channel signal; (ii) a reduced measurement noise is available for a single computation of the primitive; (iii) the attacker is able to manually preprocess a significant amount of runs of the desired primitive; (iv) the attacker attempts at averaging away the information related to the data being processed by the primitive to build the template. We note that templates (ii), (iii) and (iv) are expected to yield similar results under our assumption that the data-dependent component of the side channel signal is comparatively far smaller than the one depending on the kind of operations being computed.

We evaluated the matching capabilites of the four templates on a set of 50 traces containing 100 AES computations each, for a total of 5000 AES computations. The 5000 AES computations acted on a different key from the one employed for the templates, and fresh random plaintexts. We employed a threshold detector on the output of the matched filter to locate the time instants where an AES computation takes place, picking as thresholds values in $\{75\%, 80\%, 85\%, 90\%\}$ of the maximum value output by the filter. To prevent disturbances in this first experiment, we disable the interrupt servicing during the computation of the 100 AES.

Table 1 reports the number of detected AES computations as a function of the threshold employed. As it can be seen, employing templates (ii), (iii) and (iv) yields very similar results, in accordance with our assumption on the side channel leakage. Furthermore it is noteworthy to observe that even employing a single AES trace as a template, the detection rate remains well above 99% of the computations with an appropriately threshold. Indeed, we report that the average peak-to average output ratio of the matched filter is 1.42 for templates (ii), (iii) and (iv), and it decreases to 1.33 for template (i). This in turn justifies the perfect matching capability obtained with a threshold of 75% of the maximum value output by the matched filter.

**Table 1.** Detection capability of our approach on a sequence of undisturbed AES computations.

| Matched Filter Template | Percentage of Detected AES with Threshold | | | |
|---|---|---|---|---|
| | 75% | 80% | 85% | 90% |
| **Single trace** | 100 | 99.96 | 97.88 | 87.94 |
| **Average 1 k traces (fixed key and input)** | 100 | 100 | 100 | 99.86 |
| **Average 1 k traces (fixed key, rnd input)** | 100 | 100 | 100 | 99.88 |
| **Average 1 k traces (rnd key and input)** | 100 | 100 | 100 | 99.84 |

Processing and segmenting a one of the traces containing 100 AES computations (2.97 M samples) took around 75 seconds on our desktop. While this performance was satisfactory for our purposes, we note that the procedure can be sped up as the application of matched filters exhibits a significant amount of SIMD parallelism.

To validate the effectiveness of our approach in segmenting the side channel trace into time-aligned AES traces, we performed a Correlation Power Analysis (CPA) attack on the output of the segmentation procedure with template (iv). Correlation power analysis computes Pearson's linear correlation coefficient between a sequence of measurements of the side channel during a given operation, and a key-dependent prediction of the side channel behaviour. To obtain a computationally feasible procedure, the operation considered in CPA only involves a few key bits (e.g., 8 in our case), so that it is possible to exhaustively test for correlation all the key-dependent side channel behaviour predictions. The prediction fitting best with the actual measurements reveals the actual value of the secret key bits.

Figure 2 depicts the time-wise value of the Pearson correlation coefficient between the Hamming distance between one byte loaded as the result of the SubBytes AES primitive, and the input byte to SubBytes itself. In particular, the black line represents the correlation for the correct key hypothesis, while the grey lines pertain to incorrect key hypotheses. The noteworthy point is that only a handful of instants in time show significant correlation: we verified analyzing the assembly of the implementation that they indeed correspond to two instructions manipulating the input and output byte of the SubBytes primitive contained in the T-Table (a memory and a computational operation). This in turn points to a precise time-wise alignment of the AES traces: indeed, imperfect alignment would result in the correlation being spread over a larger time interval.

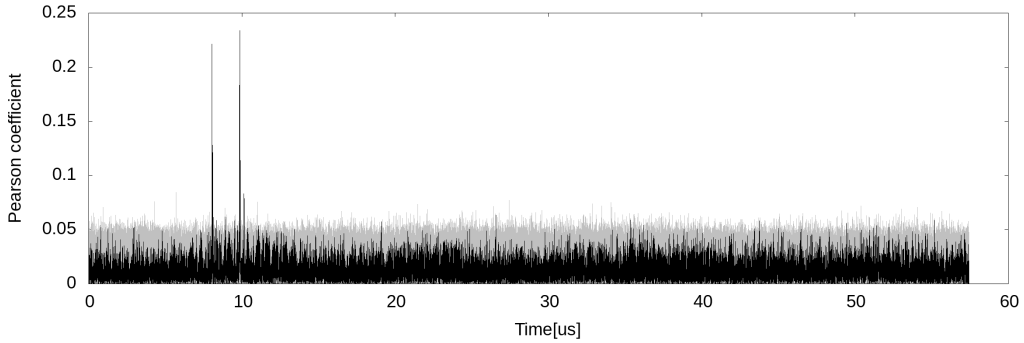
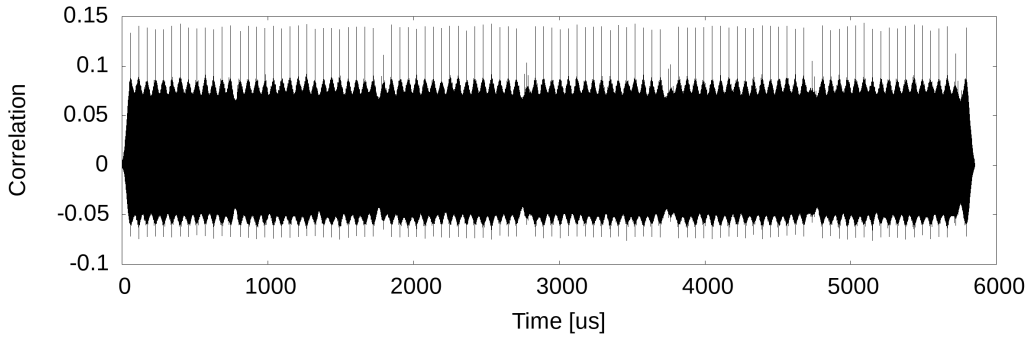


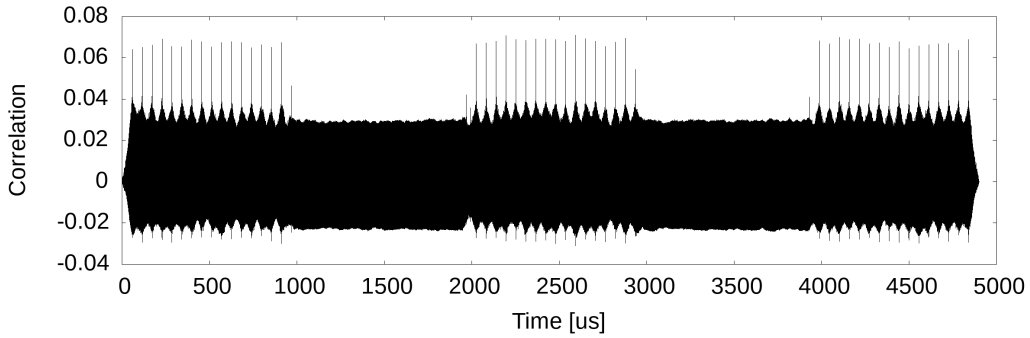

**Figure 2.** Results of a Correlation Power Analysis attack performed against a set of 5000 AES traces segmented with our matched filter approach. Sample Pearson correlation coefficient of the correct key hypothesis depicted in black, wrong key hypotheses depicted in grey.

**Identification and segmentation in presence of context changes.** After assessing the reliability of our approach in segmenting a trace containing multiple AES computations without external disturbances, we move onto assessing its reliability when the AES computations are interrupted. We consider two different scenarios: in the first we enable the interrupt servicing during the entire computation; in the second we add a second thread on the same system, performing a busy wait (i.e., computing an infinite loop) for the entirety of its time. We note that, in the first scenario, the microcontroller is not running any other

task than the AES encryption and the Miosix OS; we therefore expect that the majority of the interrupts taking place are related to the system clock signalling the end of a time slice (1 ms) assigned to a thread. In both cases we ran 100 times a thread performing 100 AES encryptions, and acting on a key different from the previous tests, and random plaintexts. Figure 3 depicts the outputs of the matched filters applied to the traces in the first scenario (Figure 3a) and in the second scenario (Figure 3b). The interruptions in the computation of the AES task, either due to the clock ISR, or to the computation of the busy wait tasks result in time periods where the matched filter detects no AES computations at all. Our automated detection and segmentation procedure detected 9452 and 9468 AES computations in the first and second scenario, respectively. This amount of detected AES computations is coherent with the fact that an AES execution interrupted by either an ISR, or the execution of the busy wait thread will be discarded by our matched filter approach. Indeed, as an AES execution takes ≈57 µs we expect that the computation of 100 AES ciphers will take 5600 µs, i.e., 5.6 ms, leading to 5.6 interruptions per set of 100 AES computations on average. The actual percentage of AES traces being discarded is 5.4% and 5.3% in the first and second scenario, respectively, which matches rather closely the expected figure.




(**a**) AES computations interrupted by interrupt service routines (roughly one interruption per ms).




(**b**) AES computations interrupted by busy waits.

**Figure 3.** Output values from the matched filter applied to a traces containing 100 AES computations being interrupted by interrupt service routines (**a**), and a second thread in busy waiting (**b**).

Figure 4 reports the outcome of a Correlation Power Analysis attack led employing the 9468 traces coming from the segmentation operated by our matched filter. We note that the Pearson Correlation has a distinct maximum time-wise, again pointing to a good alignment of the segmented traces. The lower absolute value of the correlation itself is to be ascribed to displacement in the probe positioning which is commonplace among different experiments in cases, such as ours, where no precision numerically controlled probe placement instrumentation is being employed.

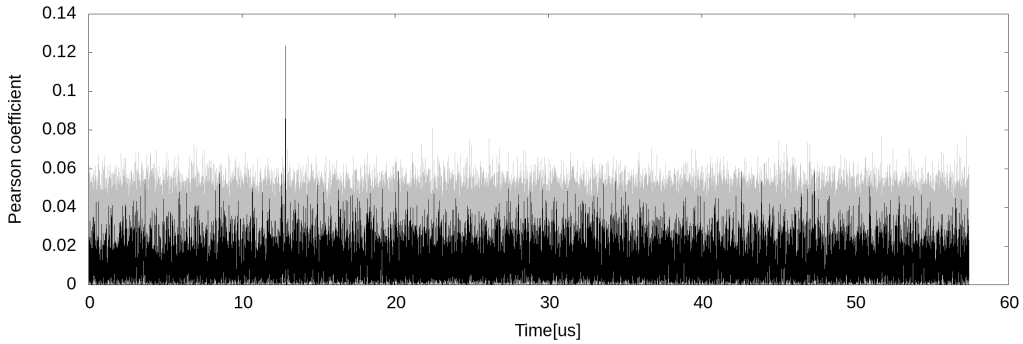



**Figure 4.** Results of a Correlation Power Analysis attack performed against a set of 9468 AES traces segmented with our matched filter approach from traces disturbed by a thread running a busy wait. Sample Pearson correlation coefficient of the correct key hypothesis is depicted in black, wrong key hypotheses is depicted in grey.

## 4. Conclusions

In this work, we presented a technique relying on the matched filter signal detection approach to locate in time the execution of a cryptographic primitive, given a recording of the side channel leakage of the device computing it. We have shown that a matched filter technique achieves perfect or near perfect detection of the AES computation by a commercial-grade Cortex-M4-based microcontroller. Our technique is computationally efficient, and can be applied to online detection, thanks to the causality of the matched filter we employed. An interesting direction for further research is to analyze the rate of potential false positives across a large software benchmark suite for microcontrollers.

**Author Contributions:** Investigation, A.B., G.F. and G.P.; Methodology, A.B. and G.P.; Software, G.F.; Writing—original draft, A.B. and G.P.; Writing—review & editing, A.B. and G.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Ronen, E.; Shamir, A.; Weingarten, A.; O'Flynn, C. IoT Goes Nuclear: Creating a Zigbee Chain Reaction. *IEEE Secur. Priv.* **2018**, *16*, 54–62. [CrossRef]
2. Randolph, M.; Diehl, W. Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman. *Cryptography* **2020**, *4*, 15. [CrossRef]
3. Moradi, A.; Barenghi, A.; Kasper, T.; Paar, C. On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from xilinx Virtex-II FPGAs. In Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, IL, USA, 17–21 October 2011; pp. 111–124. [CrossRef]
4. O'Flynn, C.; Chen, Z. Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection. *J. Cryptogr. Eng.* **2015**, *5*, 53–69. [CrossRef]
5. Balasch, J.; Gierlichs, B.; Reparaz, O.; Verbauwhede, I. DPA, Bitslicing and Masking at 1 GHz. In Proceedings of the Cryptographic Hardware and Embedded Systems-CHES 2015—17th International Workshop, Saint-Malo, France, 13–16 September 2015; Lecture Notes in Computer Science; Güneysu, T., Handschuh, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9293, pp. 599–619. [CrossRef]
6. Barenghi, A.; Pelosi, G. Side-channel security of superscalar CPUs: Evaluating the impact of micro-architectural features. In Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, 24–29 June 2018; pp. 120:1–120:6. [CrossRef]

7. Frieslaar, I.; Irwin, B. Investigating Multi-Thread Utilization as a Software Defence Mechanism Against Side Channel Attacks. In Proceedings of the 8th International Conference on Signal Processing Systems (ICSPS 2016), Auckland, New Zealand, 21–24 November 2016; pp. 189–193. [CrossRef]
8. Robyns, P.; Martino, M.D.; Giese, D.; Lamotte, W.; Quax, P.; Noubir, G. Practical operation extraction from electromagnetic leakage for side-channel analysis and reverse engineering. In Proceedings of the WiSec '20: 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Linz, Austria, 8–10 July 2020; pp. 161–172. [CrossRef]
9. Trautmann, J.; Beckers, A.; Wouters, L.; Wildermann, S.; Verbauwhede, I.; Teich, J. Semi-Automatic Locating of Cryptographic Operations in Side-Channel Traces. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2022**, *2022*, 345–366. [CrossRef]
10. Daemen, J.; Rijmen, V. *The Design of Rijndael: AES—The Advanced Encryption Standard*; Information Security and Cryptography; Springer: Berlin/Heidelberg, Germany, 2002. [CrossRef]
11. North, D. An Analysis of the factors which determine signal/noise discrimination in pulsed-carrier systems. *Proc. IEEE* **1963**, *51*, 1016–1027. [CrossRef]
12. Mangard, S.; Oswald, E.; Popp, T. *Power Analysis Attacks—Revealing the Secrets of Smart Cards*; Springer: Berlin/Heidelberg, Germany, 2007.
13. Chari, S.; Rao, J.R.; Rohatgi, P. Template Attacks. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, 13–15 August 2002; Revised Papers; Kaliski, B.S., Koç, Ç.K., Paar, C., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2523, pp. 13–28. [CrossRef]
14. STMicroelectronics. STM32-F401 Specifications. Available online: https://www.st.com/en/microcontrollers-microprocessors/stm32f401.html (accessed on 20 April 2022).
15. Collaborative Project Managed by TrustedFirmware (Formerly by Arm Ltd.). Mbed TLS: Implementation of the TLS and SSL Protocols and the Respective Cryptographic Algorithms and Support Code. 2009. Available online: https://github.com/ARMmbed/mbedtls (accessed on 20 April 2022).
16. Miosix, An OS Kernel Designed to Run on 32bit Microcontrollers. Available online: https://miosix.org/ (accessed on 20 April 2022).
17. Picoscope 5000D Series. Available online: https://www.picotech.com/download/datasheets/picoscope-5000d-series-data-sheet.pdf (accessed on 20 April 2022).