Side-Channel Protections for Cryptographic Instruction Set Extensions

Sami Saab, Pankaj Rohatgi, and Craig Hampel

Rambus Cryptography Research Division
425 Market St Fl 11
San Francisco CA 94105–2496
`{firstname}.{lastname}@cryptography.com`

Abstract. Over the past few years, the microprocessor industry has introduced accelerated cryptographic capabilities through instruction set extensions. Although powerful and resistant to side-channel analysis such as cache and timing attacks, these instructions do not implicitly protect against power-based side-channel attacks, such as DPA. This paper provides a specific example with Intel's AES-NI cryptographic instruction set extensions, detailing a DPA, along with results, showing two ways to extract AES keys by simply placing a magnetic field probe beside two capacitors on a motherboard hosting an Intel Core i7 Ivy Bridge microprocessor. Based on the insights of the DPA, methods are then presented on how to mitigate the leaks, in software, providing a dial for diverting the optimal amount of resources required for a prescribed security requirement.

Keywords: Side-Channel Analysis, DPA, Microprocessors, Cryptographic Instruction Set Extensions, Intel, AES-NI

1 Introduction

Side-channel analysis was proposed as a method of extracting cryptographic keys by Kocher [7], who noted that the time required to compute an RSA signature could reveal a private key. Further work demonstrated that one could determine cryptographic keys by observing the power consumption over time [8]. Two types of attacks were proposed. The first was inspecting a single power consumption trace, referred to as Simple Power Analysis (SPA), and a statistical treatment of a set of traces, referred to as Differential Power Analysis (DPA). It was later shown that one could use the same treatment on traces taken using electromagnetic probes [4, 10], where the equivalent attacks are typically referred to as Simple Electromagnetic Analysis (SEMA) and Differential Electromagnetic Analysis (DEMA), respectively. With adequate sampling rates, proximity, and absence of countermeasures, side-channel attacks have been practically demonstrated on a wide variety of devices ranging from small single purpose chips [2, 3, 6] to large general purpose SoCs and CPUs [5, 9, 14]. However, no analyses have exploited such a high bandwidth side channel on a commercial high-performance microprocessor such as the following results on AES-NI on an Intel Core i7. In addition, this paper also demonstrates proven and practical countermeasures that address the complexities of instruction look-ahead and instruction reordering on such devices.

In order to address the growing global demand for security in the information age, the microprocessor industry has introduced accelerated cryptographic capabilities through instruction set extensions. In addition, cryptographic instruction set extensions have been implemented on FPGAs to investigate their side-channel vulnerabilities and corresponding countermeasures [13, 12]. In this paper, we demonstrate a DPA on a commercial high-performance microprocessor; specifically Intel's AES-NI cryptographic instruction set extensions on the Intel Core i7 Ivy Bridge microprocessor. Two methods for extracting secret keys are described, the first exploits the microprocessor's cache access mechanism, exposing any implementation loading sensitive data into registers or computational units. The second exploits a sensitive state within the AES-NI instructions. We then provide methods at the Instruction Set Architecture (ISA) level that mitigate both attacks.

The remainder of this paper is organized as follows. In Section 2 we describe the Device Under Test (DUT), how power measures were performed, and how they were processed. In Section 3, we describe the first DPA and how to interpret the results to extract keys. In Section 4 we describe the second DPA and how to interpret the results to extract keys. In Section 5 we describe the DPA attack mitigations and provide code examples, revealing which leaks can be eliminated in software, and those that can not. We conclude in Section 6.

2 Test Harness and Data Collection

The DUT, an Advantech MIO-5290U-S7A1E, is a Commercial Off the Shelf (COTS) Single Board Computer (SBC) hosting an Intel Core i7 3517UE microprocessor of the Ivy Bridge microarchitecture based on Intel's 22 nm process. The microprocessor has two physical cores and supports Intel's SpeedStep Technology, Hyper-Threading, Turbo Boost 2.0, and AES-NI. The base core frequency operates at 1.7 GHz, stepping up to 2.6 GHz with both cores loaded, and 2.8 GHz with one core loaded. A full 64-bit Ubuntu 12 operating system was installed on a Solid State Drive (SSD) connected to the SBC. For the following results, Turbo Boost and Hyper-Threading were enabled and core switching was left under OS control.

An application was written in C, making calls to Intel's AES-NI Sample Library v1.2 [1]. The application was written to loop over a group of AES-256 encrypts in CBC mode by making calls to the assembly routine `iEnc256_CBC`. Initially, the application was configured to run continuously, alternating between encrypting a large block of random data and sleeping for a specified block of time. Concurrently, various probes and locations were used to find emanating electromagnetic radiation that correlated to when the microprocessor was encrypting. The Langer magnetic field probe RF-U 2,5-2 and the location as shown in Fig. 1

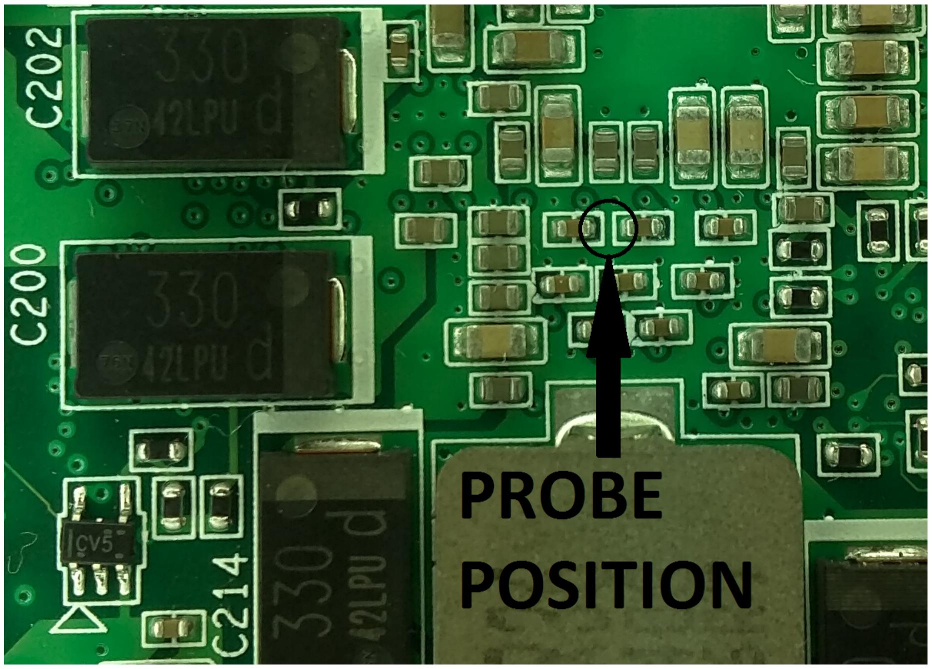

Fig. 1. Probe position between two small capacitors on the back of the SBC.

provided a good power signature that correlated to various microprocessor activities. The resulting signal also exhibited a low frequency drift that did not correlate to microprocessor activity and so Mini-Circuits ZX60-33LN-SS+ amplifiers with a low frequency cutoff of 50 MHz were used. Due to the signal's low power, two amplifiers were connected in series to amplify the signal by approximately 40 dB before being sampled by an oscilloscope.

AES-NI consists of the seven instructions summarized in Table 1. `AESDEC`, `AESDECLAST`, `AESENC`, and `AESENCLAST` take two parameters: a 128-bit round input state and a 128-bit round key. With AVX support, a third parameter can be provided to receive the round output state; otherwise, the round input state is overwritten by the round output state. Different architectures provide different throughputs and latencies with the Ivy Bridge microarchitecture producing single cycle throughput and eight cycle latency. Based on the DUT's core clock rate, AES-NI instruction throughput, and number of encryptions per loop, an estimated time for encrypting one block of data was calculated. Patterns of similar duration were then searched for in the power signal. Fig. 2 presents two power measurement traces, each taken under different operating parameters. Each trace exhibits a periodic section of higher energy. Inspecting the library routine `iEnc256_CBC`, one can attribute the higher energy to initialization and alignment of round keys in memory. As we could not find any features in the trace caused by the encryption directly, we used these high energy features to

Table 1. AES-NI Instructions

Instruction	Operation
AESDEC	one round of AES decryption
AESDECLAST	last round of AES decryption
AESENC	one round of AES encryption
AESENCLAST	last round of AES encryption
AESIMC	inverse mix columns
AESKEYGENASSIST	part of round key generation
PCLMULQDQ	64-bit carry-less multiply

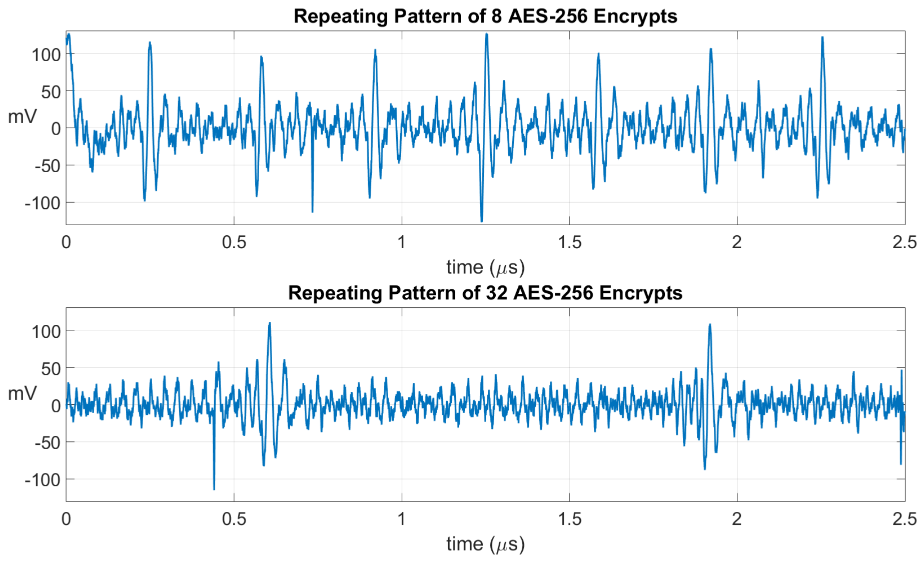

Fig. 2. Two power measurement traces sampled under different operating parameters. Each trace displays delimiters of higher power with periods that correlate to the predicted time to process the specified amount of data.

synchronously trigger the scope with each block of encryption. We also reduced some noise from each collected trace by looping the same values in every data block and averaging on the scope.

3 Input Analysis and Interpretation

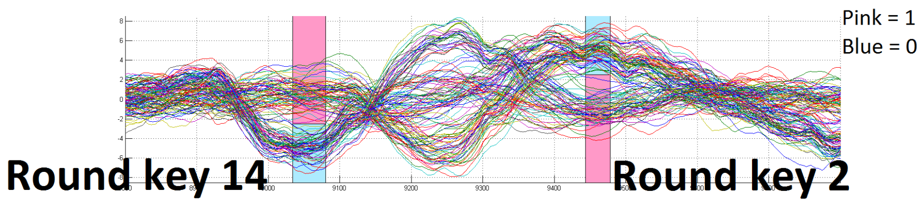

Fig. 3. Input analysis revealing Hamming distance leaks between alternating loads from cache.

After collecting 1,500,000 power measurement traces over a period of 17 days, each an average of 256 blocks of eight encrypts, and performing some alignment and Wavelet based denoising [11], the variance normalized difference of means based on the value of each of the 128 bits of input yielded the statistically significant results shown in Fig. 3. Moreover, the polarity of the difference of means for each bit provided a direct relationship with two other values; specifically the second and second to last round keys. Inspecting the library routine `iEnc256_CBC`, one can explain this relationship as a power dependency between alternating loads from cache.

Table 2. Input Analysis Results on a Hamming Distance Leak

Fixed Secret	**Changing Inputs**			Polarity from
	1s	—	**0s**	measured HD
Transitions between Fixed and **Changing** data				
1	1 & **1s**	—	1 & **0s**	$0-1 \rightarrow \downarrow$
0	0 & **1s**	—	0 & **0s**	$1-0 \rightarrow \uparrow$

Generally, when considering three 128-bit values loaded from cache, if a transition occurs at a bit position between the first and third value, the microprocessor will emanate more energy than if a transition does not occur. This is true regardless of the number of instructions between any of the loads and the value of the second load. If sorting power measurements based on bit values of either the first or third load, and subtracting the mean of power measurements associated with a zero value from the mean of power measurements associated with a one value, the polarity of the resulting difference of means will be negative if the associated bit from the third or first load respectively is always one, and positive if always zero, as shown in Table 2. As the changing input in addition to the fixed round keys are loaded from cache, this power dependency manifests itself between the input and two loads later (the second round key), and two loads earlier from the previous block of encrypts (the second to last round key). This polarity leak on the input analysis breaks AES-128 and provides half of the secret key for AES-256.

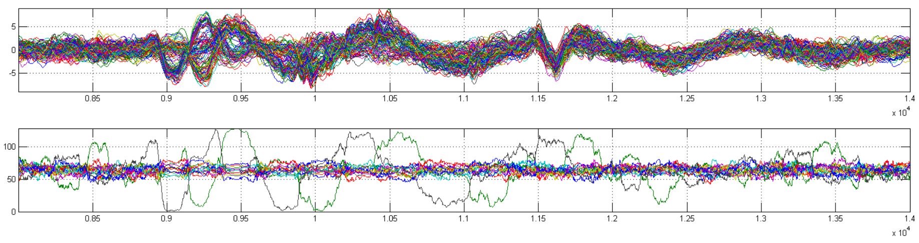

Fig. 4. Top plot shows input analysis difference of means over a longer time window. Bottom plot shows the number of correctly guessed bits for each round key. At each point in time, all 128 bits are guessed to be 1 if their difference of means is less than the mean of all 128 difference of means. The number of guessed bits that are correct for each round key are then plotted. Only round keys 2 and 14, green and black lines respectively, show any correlation, that in turn manifest as a sinusoidal decay in polarity.

Note however the reverse of the expected polarity for round key 14 in Fig. 3. Clearly other events are occurring within the microprocessor that depend on the same relationship. Other phenomena also at play include how the power is measured. The power signal that originates within the microprocessor can be approximated as an impulse response, and is filtered by the packaging, bypass capacitors, probe, amplifier and oscilloscope front end before being sampled. As such, a single event will manifest in a measured power trace as this implicit filter's transfer function. Fig. 4 shows a zoomed out region of the difference of means from Fig. 3. As can be seen, the polarity alternates for both round keys, exhibiting the pattern of an impulse response passing through a filter.

4 Round Hamming Distance Analysis and Interpretation

Although the input analysis is serious, it can be mostly prevented without much processing overhead. It is also worth noting that the input leak is not directly related to the AES-NI. However, there is another leak that is directly related to the AES-NI and is not as easy to mitigate.

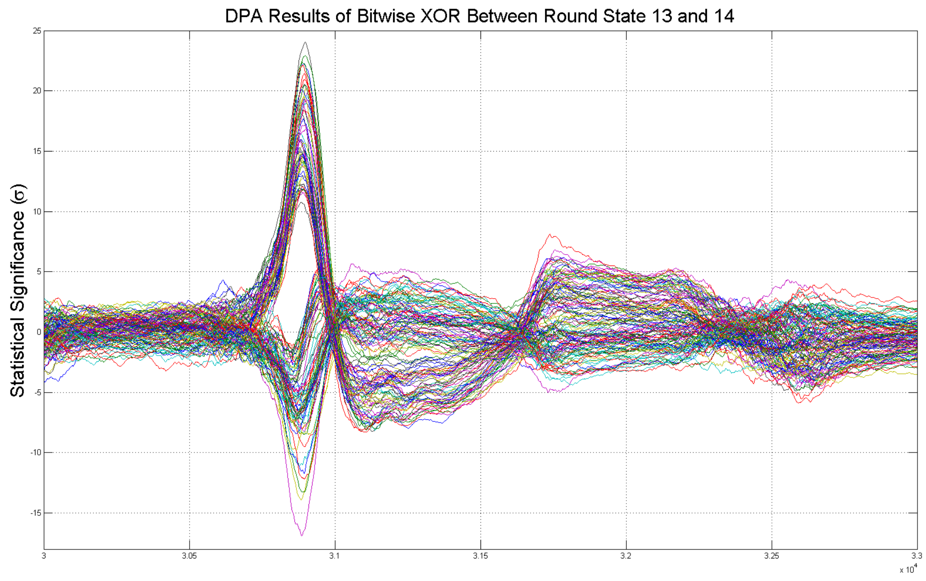

Fig. 5. Hamming distance between round input state and round output state analysis revealing Hamming distance leaks between state in previous round before the XOR of the round key and current round before the XOR of the round key.

After collecting 1,341,165 power measurement traces over a period of 22 days, each an average of 256 blocks of 32 encrypts, the variance normalized difference of means based on the Hamming distance between the start and end of a round for each of the 128 bits of state yielded the statistically significant results as shown in Fig. 5. Moreover, the polarity of the difference of means for each bit provided a direct relationship with another value; specifically the round key associated with the previous round.

Table 3. Hamming Distance (HD) Between Round States (S) Results on a HD Leak Before Mixing of Round Keys (RKs)

$\underline{\mathrm{RK}_N}$	$\mathbf{RK}_{N+1}$	S_{N+1}	S_{N+2}		S_{N+1}	S_{N+2}	Polarity from
		HD = 1		—	HD = 0		measured HD
$\underline{1}$	$\mathbf{1}$	$\underline{1} \oplus 1$	$\mathbf{1} \oplus 0$	—	$\underline{1} \oplus 1$	$\mathbf{1} \oplus 1$	$0 \oplus 1 - 0 \oplus 0 \rightarrow \uparrow$
$\underline{1}$	$\mathbf{1}$	$\underline{1} \oplus 0$	$\mathbf{1} \oplus 1$	—	$\underline{1} \oplus 0$	$\mathbf{1} \oplus 0$	$1 \oplus 0 - 1 \oplus 1 \rightarrow \uparrow$
$\underline{1}$	$\mathbf{0}$	$\underline{1} \oplus 1$	$\mathbf{0} \oplus 0$	—	$\underline{1} \oplus 1$	$\mathbf{0} \oplus 1$	$0 \oplus 0 - 0 \oplus 1 \rightarrow \downarrow$
$\underline{1}$	$\mathbf{0}$	$\underline{1} \oplus 0$	$\mathbf{0} \oplus 1$	—	$\underline{1} \oplus 0$	$\mathbf{0} \oplus 0$	$1 \oplus 1 - 1 \oplus 0 \rightarrow \downarrow$
$\underline{0}$	$\mathbf{1}$	$\underline{0} \oplus 1$	$\mathbf{1} \oplus 0$	—	$\underline{0} \oplus 1$	$\mathbf{1} \oplus 1$	$1 \oplus 1 - 1 \oplus 0 \rightarrow \downarrow$
$\underline{0}$	$\mathbf{1}$	$\underline{0} \oplus 0$	$\mathbf{1} \oplus 1$	—	$\underline{0} \oplus 0$	$\mathbf{1} \oplus 0$	$0 \oplus 0 - 0 \oplus 1 \rightarrow \downarrow$
$\underline{0}$	$\mathbf{0}$	$\underline{0} \oplus 1$	$\mathbf{0} \oplus 0$	—	$\underline{0} \oplus 1$	$\mathbf{0} \oplus 1$	$1 \oplus 0 - 1 \oplus 1 \rightarrow \uparrow$
$\underline{0}$	$\mathbf{0}$	$\underline{0} \oplus 0$	$\mathbf{0} \oplus 1$	—	$\underline{0} \oplus 0$	$\mathbf{0} \oplus 0$	$0 \oplus 1 - 0 \oplus 0 \rightarrow \uparrow$

Indeed, just as with the input analysis, a bit transition emanates more power than no bit transition. However, the Hamming distance between the states that leak and the states measured have a linear relationship with their neighboring round keys. The actual Hamming distance leak resides before the mixing of the round key between successive rounds. Therefore, the function between the measured states and the leaking states is an XOR with the round keys of their previous rounds respectively. If sorting power measurements based on the Hamming distances between a bit in the states at the start and end of a round as follows,

$$\begin{aligned}
\mathrm{Set}_A &= \mathrm{AVG}(\text{measurements associated with } HD = 1) \\
\mathrm{Set}_B &= \mathrm{AVG}(\text{measurements associated with } HD = 0) \\
\mathrm{D}_{AB} &= \mathrm{Set}_A - \mathrm{Set}_B
\end{aligned}$$

then, $\mathrm{D}_{AB} < 0$ if the associated bits between the round keys are different, and $\mathrm{D}_{AB} > 0$ if the associated bits between the round keys are the same, as shown in Table 3.

Note that to break AES-128, one does not need to account for this relationship in order to extract one round key. However, one can use this relationship to break AES-256, extracting two successive round keys by analyzing just one round. In addition, if attacking the last round of AES, the search space for breaking AES-256 is only 16×2^8. Following is the specific attack procedure for breaking AES-256 from the last round.

```
– For each 16, 8-bit sub keys
   – Compute HD of 8 state bits from all sub key guesses
   – For each guess
   –– Compute AVG[ABS(difference of means traces)]
   – Set guess with MAX result as correct sub key value
– Record polarity of each correct difference of means trace
   – XOR resulting 128-bit value with $RK_{15}$ to get $RK_{14}$
```

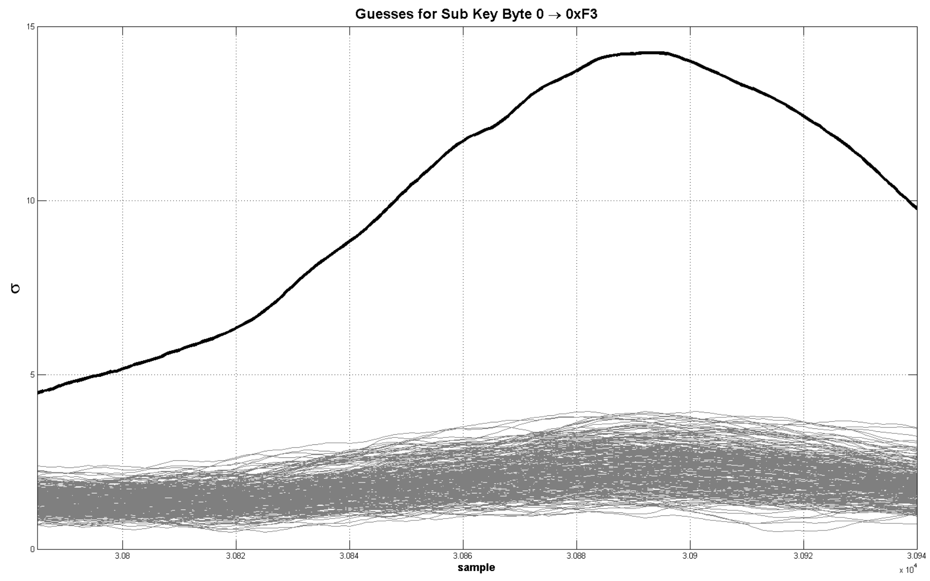

Fig. 6. Average of absolute of difference of means traces for all possible values of a single sub key. The highest trace in black corresponds to the correct sub key guess.

Fig. 6 shows a set of AVG[ABS(difference of means traces)] for all guesses of a single sub key.

5 Leak Mitigation

The reason for the input analysis leak is unknown, but hypothesized to be sequential 128-bit loads from cache alternating between the lower and upper halves of a 256-bit internal state register. As the DUT's microprocessor supports AVX instructions, it is reasonable to assume such 256-bit registers exist, and such a construct would explain why only alternating loads from cache interact with one another. As mentioned at the beginning of the previous section, the input analysis leak can be mostly prevented without too much processing overhead. For instance, one could insert two dummy loads before and after each load of sensitive data. Better yet, to minimize performance overhead one could insert two dummy loads before and after a collection of sensitive data loads and around any loads of known changing data, such as input data. Care must be taken to ensure dependencies between dummy loads so as to prevent the microprocessor's scoreboarding logic from reordering the loads. This technique for the most part eliminates the input analysis leak. The only caveat would be if another program were to interrupt the process with known changing loads either before or after fixed secret loads.

The following macro can be used to insert two dummy loads before loading a changing known value or a fixed secret value as the second argument (%2) and computing its XOR with the first argument (%1).

```
%macro dload_pxor 2
   pxor %1, [pad + 0]
   pxor %1, [pad + 16]
   pxor %1, %2
%endmacro
```

Understanding and addressing the round Hamming distance analysis leak is not as straight forward and can incur noticeable performance overhead. Note that using the AVX variant of the AESENC and AESENCLAST instructions by assigning the round output state into a different register than the round input state does not work as the leak resides within the instructions themselves and not when overwriting the result to a register specified at the ISA level. The first test to determine the origin of this leak involved using the PXOR instruction. As the location of the state that leaks resides before mixing the round key, the thought was that if the AESENC and AESENCLAST instructions shared the XOR logic with the PXOR instruction, then the Hamming distance relationship between successive AESENC and AESENCLAST instructions could be broken by inserting a dummy PXOR instruction in between. However, DPA showed this not to be the case. The final solution came by inserting dummy AESENC and AESENCLAST instructions in between. Note that due to the pipelining capabilities of the AESENC and AESENCLAST instructions, this solution does not slow down the throughput of AES encrypts until the number of encrypts performed in parallel fill the pipe. Again, care must be taken to ensure that the microprocessor's scoreboarding logic does not reorder the strategically placed dummy operations intended to break relationships between states over adjacent rounds. The final solution involved a separate independent stream of dummy AESENC and AESENCLAST instructions, with a random input seed and 15 random round key seeds. Each dummy round assigned its output round state to the round key used for the next dummy round.

The following macros and code snippet show how the execution order was fixed with required dependencies.

```
%macro aesencx 4
   aesenc %1, %2
   aesenc %3, %4
   movdqa %4, %3
%endmacro


%macro aesencxlast 4
   aesenclast %1, %2
   aesenclast %3, %4
   movdqa     %4, %3
%endmacro


; [rk]  points to secret round keys
; [mix] points to dummy  round keys
aesencx xmm0, [rk+1*16], xmm1, [mix+0*16]
```

Using this approach, the Hamming distance leak was successfully removed. However, further analysis showed that after 75 times more traces, the leak reappeared, presumably based on the Hamming weight of the state before the mixing of the round key. Unfortunately, this leak cannot be directly addressed at the ISA level due the state register not being directly accessible.

6 Conclusions

Assuming power based side-channel analysis is only relevant with low-power devices is a common misconception. As has been shown, power based side-channel analysis is just as relevant with large devices [5, 9, 14]. As has been shown in this paper, very large and complicated microprocessors in a complex environment running full and complex operating systems are also susceptible. The good news is, in this case, practical countermeasures can be introduced in software to increase resistance and meet many security requirements. However, as security becomes more important in the emerging IoT age, such vulnerabilities are growing in relevancy and need to be acknowledged and addressed.

References

1. Intel aesni sample library (May 2011), `https://software.intel.com/en-us/articles/download-the-intel-aesni-sample-library`

2. Balasch, J., Gierlichs, B., Verdult, R., Batina, L., Verbauwhede, I.: Power analysis of Atmel CryptoMemory – recovering secret keys from secure EEPROMS. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178. Springer
3. Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., Shalmani, M.M.: On the power of power analysis in the real world: A complete break of the KeeLoq code hopping scheme. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157. Springer
4. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, C.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer (2001)
5. Genkin, D., Pipman, I., Tromer, E.: Get your hands off my laptop: Physical side-channel key-extraction attacks on pcs. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 242–260. Springer (2014)
6. Kasper, M., Kasper, T., Moradi, A., Paar, C.: Breaking KeeLoq in a flash: On extracting keys at lightning speed. In: Preneel, B. (ed.) AFRICACRYPT. Springer
7. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO '96. LNCS, vol. 1109, pp. 104–113. Springer (1996)
8. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO '99. LNCS, vol. 1666, pp. 388–397. Springer (1999)
9. 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: CCS. pp. 111–124 (2011)
10. Quisquater, J.J., Samyde, D.: Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In: Attali, I., Jensen, T.P. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer (2001)
11. Sami Saab, A.L., Tunstall, M.: Efficient key extraction from the primary side of a switched-mode power supply. Cryptology ePrint Archive, Report 2015/512 (2015), `http://eprint.iacr.org/`
12. Tillich, S., Groschdl, J.: Power analysis resistant aes implementation with instruction set extensions. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2007, Lecture Notes in Computer Science, vol. 4727, pp. 303–319. Springer Berlin Heidelberg (2007), `http://dx.doi.org/10.1007/978-3-540-74735-2_21`
13. Tillich, S., Herbst, C., Mangard, S.: Protecting aes software implementations on 32-bit processors against power analysis. In: In Applied Cryptography and Network Security ACNS 2007. p. 141
14. Uno, H., Endo, S., Hayashi, Y., Homma, N., Aoki, T.: Chosen-message electromagnetic analysis against cryptographic software on embedded OS. In: EMC '14. pp. 314–317. IEEE (2014)