Tackling the Time-Defence: An Instruction Count Based Micro-Architectural Side-Channel Attack on Block Ciphers

Manaar Alam
Sarani Bhattacharya
Debdeep Mukhopadhyay

Indian Institute of Technology Kharagpur

SPACE 2017
December 17, 2017

Objective of Talk

- Investigating the presence of information leakage of an encryption process using the hardware event *instruction* count.

- Our Contributions:
  - We propose Instruction Profiling Attack (IPA), tailored for block ciphers using HPC event: *instruction*. Effectiveness of IPA is to successfully retrieve the secret key bits.
  - We evaluate our proposed attack method on both Intel and AMD platforms. The time complexity of successful IPA is much lesser than a successful cache-timing attack.
  - Additionally, we show success rate of IPA is not affected by time-obfuscation countermeasures like TimeWarp.

Time obfuscation countermeasures

- Profiling and analysis of timing information uses the Time-Stamp Counter (TSC).
- Common countermeasure to thwart attacks using timing channel is to provide the adversary a modified timing information instead of the real one.
- The obfuscation of RDTSC [1] can be done in two ways, first by introducing the concept of the real offset, which is the insertion of a real-time delay that stalls RDTSC execution, and then using the apparent offset that is by modification of the return value of the instruction by a small amount.

Hardware Performance Counters

- HPCs are a set of special purpose registers, which are present in most of the modern microprocessor's Performance Monitoring Unit (PMU).
- These registers store hardware and software events related to the execution of a program, such as cache misses, retired instructions, retired branch instructions, and so on.
- Every popular operating systems have HPC-based profilers, type and number of hardware events vary across different Instruction Set Architectures (ISA) [2].

Metrics of Evaluation

Success Rate The o^{th} order *success rate* of the side channel attack $A_{E_K,L}$ against the key classes is defined as [3]:

$$Succ^{o}_{A_{E_K},L}(\tau, m, k) = Pr[Exp_{A_{E_K,L}} = 1]$$

Guessing Entropy The Guessing Entropy of the adversary $A_{E_k,L}$ is defined as [3]:

$$GE_{A_{E_K,L}}(\tau, m, k) = E[Exp_{A_{E_K,L}}]$$

Outline

1. Information Leakage due to Instruction Count
2. Description of IPA
3. Performance Evaluation of IPA

1. Information Leakage due to Instruction Count

2. Description of IPA

3. Performance Evaluation of IPA

Performance monitoring tools

perf: Capable of statistical profiling of the entire system by instrumenting the hardware performance counters. *perf* supports a list of hardware events to monitor, like cache-misses, branch-misses, cpu-cycles, etc. For our proposed attack we observe the event `instruction` to analyze the source of side-channel.

```
perf stat -e instructions ./aes <plaintext>
```

PAPI: More sophisticated than perf tool since it provides a larger number of hardware events for monitoring, like `PAPI_BR_INS` can be used to measure the total branch instructions whereas `PAPI_LST_INS` can be used to measure the total load/store instructions executed for an encryption algorithm.

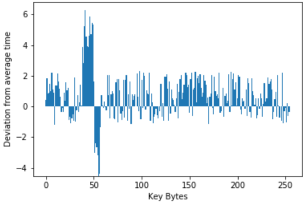
Correlation of Cache events to Instruction Counts

Cache timing attacks exploit the non-uniformity in access times of the table lookup requests in block-ciphers, which are typically attributed to cache memory events such as cache hit and miss.

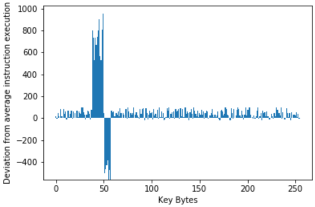
- On a cache miss, a memory element from the cache is written back in the main memory, followed by loading of a new data element in the particular location of the cache. Hence, this event is inherently performed with a higher number of instructions.
- On a cache hit, the processor will not issue any additional instructions to load the data from the memory as the required data is already present in the cache.

Conclusion: The cache events have an alternative effect on the instruction count event.

Profiling the Instruction counts



(a) Deviation of total encryption time from average



(b) Deviation of total instructions executed during an encryption from average

Experiment: Plaintext byte p_0 is varied randomly from 0 to 255, keeping all other bytes unchanged for a fixed secret key on the OpenSSL AES Encryption.

- The timing profile using RDTSC instructions for the key-byte k_0 is shown in Fig. (a)
- The instruction profile for the key-byte k_0 is shown in Fig. (b)

Conclusion: Average deviation for instruction counts have a similar profile as that of the timing profile constructed for cache timing attacks.

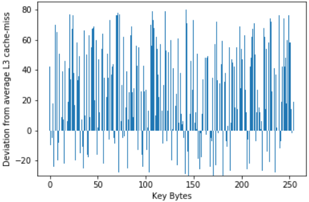


Figure: Deviation of total L3 cache-misses during an encryption from average

We replicated the same experiment replacing instruction count with cache misses.

- The cache miss profile is shown in the Figure.
- The cache miss event monitored through PAPI observes the cache misses from all the three levels of cache, which is highly noisy as in the figure and bears no resemblance to the timing observation.

Conclusion: Instruction count bears a direct correspondence to the timing side channel rather than the event cache misses.

Analyzing Load/store instruction counts

We explore different types of instructions using the tool PAPI to investigate the type of instructions responsible for generating similar profile as timing.

Data Manipulation Instructions

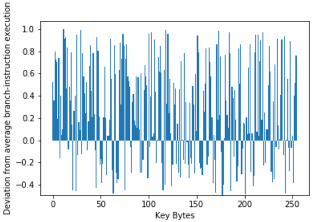
- Consists of arithmetic instructions, logical instructions, etc. Provides computational power to the computer by performing different operations on data.
- The hardware events for monitoring these instructions are `PAPI_INT_INS`, `PAPI_FP_INS`, etc., which measure the total integer instructions, total floating point instructions respectively, spawned by the processor. However, the PAPI tool does not provide the handle to observe these hardware events.

Data Transfer Instructions

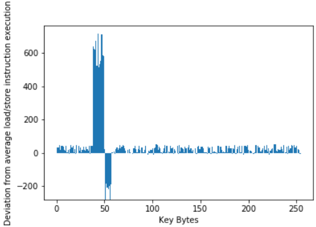
- Transfer data between memory and registers, register & input or output, and between processes register without changing the data content.
- The handle to monitor these instructions is the hardware event `PAPI_LST_INS`.

Program Control Instructions

- Direct or change the flow of a program. Consists of all the branch instructions.
- We can observe these instructions by using the hardware event `PAPI_BR_INS`.



(a) Deviation of total branch instructions executed during an encryption from average



(b) Deviation of load/store instructions executed during an encryption from average

Experiment: Plaintext byte p_0 is varied randomly from 0 to 255, keeping all other bytes unchanged for a fixed secret key on the OpenSSL AES Encryption.

- The profile using `PAPI_BR_INS` instructions for the key-byte k_0 is shown in Fig. (a)
- The profile using `PAPI_LST_INS` instructions for the same key-byte k_0 is shown in Fig. (b)

Conclusion: The load and store instructions bear a direct resemblance to the timing characteristics and thus are a potential source of leakages.

1. Information Leakage due to Instruction Count

2. Description of IPA

3. Performance Evaluation of IPA

The proposed attack consists of three different phases: *Offline Profiling*, *Online Attack*, and *Correlation*.

Offline Profiling We generate a set of random plaintext $P = \{p_0, p_1, \cdots, p_l\}$ and observe the total instruction count for each encryption with a known secret key k, which can be written as $ins(E_{AES}(p_i, k))$. We store the average instruction count for each byte and for each value of that byte in a matrix $I[16][256]$. For each plaintext p_i $(0 \leq i \leq l)$ and for each byte $(0 \leq j \leq 15)$ we successively compute the elements of the matrix $I[16][256]$ as below:

$$I[j][p_{j,i}] = I[j][p_{j,i}] + ins(E_{AES}(p_i, k))$$

where, $p_{j,i}$ is the j^{th} byte of the i^{th} plaintext. We have for $0 \leq j \leq 16$ and $x \in \{0, 1, \cdots, 255\}$

$$I[j][x] = \sum_{\{i|p_{j,i}=x\}} ins(E_{AES}(p_i, k))$$

We then calculate the average number of instructions taken by each byte for each value of that byte using

$$v[j][y] = \frac{I[j][y]}{num[j][y]} - \frac{\sum_j \sum_x I[j][x]}{\sum_j \sum_x num[j][x]}$$

where, $num[16][256]$ stores the total number of measurements per value of a byte value.

Online Attack We generate a random set of plaintexts $P' = \{p'_0, p'_1, \cdots, p'_l\}$. We follow the same approach as discussed in profiling phase and calculate two matrices I' and num' for the unknown key k'. We then calculate the matrix v' as defined before.

Correlation We correlate the two matrices v and v' obtained from the previous steps. A matrix $c[16][256]$ is created using the following definition for $0 \leq j \leq 15$ and $0 \leq u \leq 255$.

$$c[j][u] = \sum_{w=0}^{255} v[j][w] \cdot v'[j][u \oplus w]$$

The elements of the matrix c are then sorted in decreasing order for each row. The highest correlated key value for a particular byte is the candidate key for that key byte. The full secret key can be recovered by *Brute Force* search for the remaining secret key bytes with very narrow search space.

1 Information Leakage due to Instruction Count

2 Description of IPA

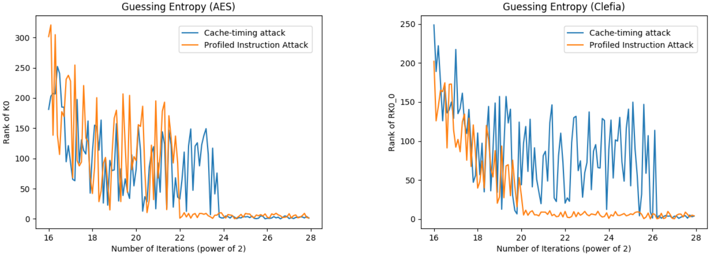
3 Performance Evaluation of IPA

Experimental Setup

- We validated the proposed technique on two different environments to get a generalized performance measure.

1.	*Setup 1*	Intel Core i5-4570 CPU with 3.20GHz clock frequency, 256KB of L1 Data Cache, 1MB L2 Data Cache, 6MB L3 Data Cache.
2.	*Setup 2*	AMD A10-8700P CPU with 1.8GHz clock frequency, 320KB of L1 Data Cache, 2MB of L2 Data Cache.

More Efficient than Timing Attacks [4, 5]



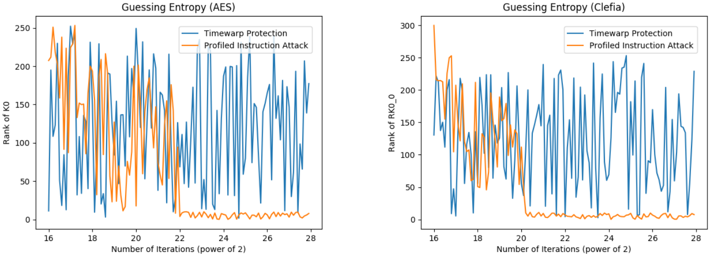
(c) Predicting first key-byte k_0 of AES

(d) Predicting round key-byte $RK0\_0$ of Clefia

Figure: Comparing Guessing Entropy of IPA with respect to cache-timing attack for Setup 1

- Cache-timing attack needs 2^{24} iterations to correctly predict k_0 of AES and 2^{26} iterations for $RK0\_0$ of Clefia.
- Whereas IPA requires 2^{22} and 2^{20} iterations respectively.

Efficient in Presence of TimeWarp [1]



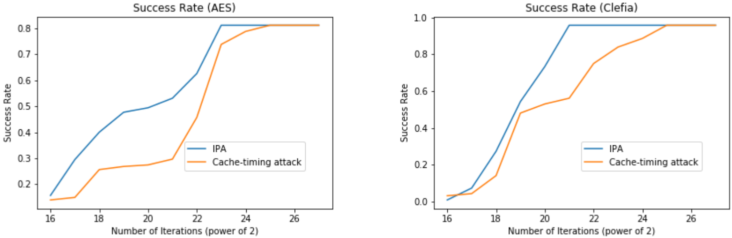
(a) Predicting first key-byte k_0 of AES

(b) Predicting round key-byte $RK0\_0$ of Clefia

Figure: Comparing Guessing Entropy of IPA with respect to cache-timing attack for Setup 1 in presence of TimeWarp implementation

- Cache-timing attack can not predict k_0 and $RK0\_0$ even after 2^{28} iterations.
- Whereas time obfuscation does not have any impact on the performance of IPA.

Success rate of the proposed IPA



(a) Success rate plot of IPA and Cache-timing attack in predicting secret key-bytes of AES

(b) Success rate plot of IPA and Cache-timing attack in predicting secret key-bytes of Clefia

Figure: Comparison of Success Rate of IPA with Cache-timing attack in Setup 1

The success rate of IPA is better than the classical timing attacks for both the ciphers.

Retrieving the Secret Key

Table: Correctly retrieving secret of AES using IPA with Timewarp countermeasure

		k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}	k_{11}	k_{12}	k_{13}	k_{14}	k_{15}
	Actual Key	e2	6d	b3	f5	64	42	c6	92	3e	0f	96	b6	a1	52	9d	86
Predicted Key	Setup 1	**e2**	**6d**	a4	**f5**	**64**	**42**	**c6**	**92**	3a	**0f**	**96**	ae	**a1**	**52**	**9d**	**86**
	Setup 2	**e2**	**6d**	**b3**	**f5**	**64**	b8	**c6**	**92**	**3e**	**0f**	**96**	34	**a1**	**52**	**9d**	**86**

Table: Failure in retrieving secret of AES using timing channel with Timewarp countermeasure

		k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}	k_{11}	k_{12}	k_{13}	k_{14}	k_{15}
	Actual Key	e2	6d	b3	f5	64	42	c6	92	3e	0f	96	b6	a1	52	9d	86
Predicted Key	Setup 1	d9	ea	b3	24	4d	08	6d	64	4f	fc	c2	6d	af	dc	64	88
	Setup 2	72	8a	3b	6e	7f	4c	dc	1c	7f	42	af	a6	76	20	7c	b2

Table: Retrieving Round Keys RK0 for Clefia in Setup 1 using total instruction count with Timewarp countermeasure

	Top-4 probable RK0 Round Keys (Correlation Value)			
RK0_0	**0xbe (250.167)**	0x6b (30.158)	0x7e (31.148)	0xee (23.137)
RK0_1	**0xf8 (260.326)**	0xd1 (34.242)	0xfc (33.682)	0xe8 (31.024)
RK0_2	**0xe7 (255.388)**	0x9f (57.648)	0x31 (56.957)	0x7a (54.515)
RK0_3	**0xae (255.851)**	0x87 (33.130)	0xbe (32.455)	0x41 (30.312)

Table: Retrieving Round Keys RK0 for Clefia in Setup 2 using total instruction count with Timewarp countermeasure

	Top-4 probable RK0 Round Keys (Correlation Value)			
RK0_0	**0xbe (260.166)**	0x6b (34.153)	0x7e (31.147)	0xee (30.130)
RK0_1	**0xf8 (265.456)**	0xd1 (33.478)	0xfc (32.147)	0xe8 (31.200)
RK0_2	**0xe7 (247.457)**	0xae (57.124)	0x43 (57.008)	0x95 (54.214)
RK0_3	**0xae (259.567)**	0x87 (47.247)	0xbe (46.211)	0x41 (40.589)

Table: Failure in retrieving Round Keys RK0 for Clefia in Setup 1 using timing attack with Timewarp countermeasure

	Top-4 probable RK0 Round Keys (Correlation Value)			
RK0_0	0xce (314.699)	0x65 (289.491)	0x20 (276.232)	0xae (213.873)
RK0_1	0xac (775.449)	0x68 (761.411)	0xbb (603.751)	0xb2 (577.428)
RK0_2	0x56 (453.751)	0xd7 (417.697)	0xc8 (347.645)	0xfe (249.147)
RK0_3	0x37 (598.248)	0xac (548.479)	0x6b (497.268)	0xd5 (457.314)

Table: Failure in retrieving Round Keys RK0 for Clefia in Setup 2 using timing attack with Timewarp countermeasure

	Top-4 probable RK0 Round Keys (Correlation Value)			
RK0_0	0xd7 (478.324)	0xba (421.984)	0x2e (394.157)	0xcf (350.496)
RK0_1	0x1e (367.459)	0xf9 (314.496)	0xa1 (296.549)	0xd9 (247.693)
RK0_2	0x8a (724.967)	0x4d (695.349)	0xa0 (645.945)	0x09 (634.235)
RK0_3	0xe4 (676.935)	0x45 (645.453)	0x00 (601.239)	0xda (509.486)

Table: Retrieving all the Round Keys of Clefia

Round Key Bytes	Correct Key	Predicted Key (Setup 1)	Predicted Key (Setup 2)
RK0_0	be	**be** (250.548) 6b (33.518)	**be** (331.478) 72 (30.347)
RK0_1	f8	**f8** (236.478) d1 (34.398)	**f8** (347.149) 0e (32.478)
RK0_2	e7	**e7** (259.496) 9f (31.759)	**e7** (312.479) e9 (35.478)
RK0_3	ae	**ae** (249.247) 87 (30.974)	**ae** (299.647) 5e (31.479)
RK1_0	75	**75** (239.479) 6e (29.647)	**75** (357.457) 14 (29.475)
RK1_1	61	**61** (213.795) 0a (37.198)	**61** (378.147) 2d (47.149)
RK1_2	b8	**b8** (297.347) 54 (30.789)	**b8** (249.647) 64 (36.759)
RK1_3	30	**30** (267.126) 7c (31.496)	**30** (432.148) 1f (26.478)
RK2_0 + WK0_0	91	**91** (257.214) d3 (34.189)	**91** (496.487) 77 (31.478)
RK2_1 + WK0_1	e1	**e1** (269.147) b6 (33.698)	**e1** (249.657) 36 (32.768)
RK2_2 + WK0_2	3e	**3e** (259.347) fb (31.478)	**3e** (387.162) 32 (26.479)
RK2_3 + WK0_3	46	**46** (249.347) df (29.678)	**46** (321.338) 3c (47.147)
RK3_0 + WK1_0	34	**34** (298.147) 87 (35.148)	**34** (410.814) cc (43.549)
RK3_1 + WK1_1	1f	**1f** (267.348) 8d (31.987)	**1f** (490.703) c6 (29.647)
RK3_2 + WK1_2	5f	**5f** (249.347) ff (34.158)	**5f** (228.757) f9 (33.679)
RK3_3 + WK1_3	6f	**6f** (219.347) 38 (36.489)	**6f** (353.479) f2 (29.624)
RK4_0	70	**70** (278.498) 1a (32.489)	**70** (228.749) 7e (45.697)
RK4_1	c7	**c7** (264.369) b0 (29.634)	**c7** (249.647) 53 (49.547)
RK4_2	cc	**cc** (249.149) 66 (34.214)	**cc** (349.248) 04 (23.452)
RK4_3	d8	**d8** (278.694) 24 (28.365)	**d8** (324.479) 2e (32.479)
RK5_0	b8	**b8** (324.496) 68 (49.324)	**b8** (367.457) 33 (36.139)
RK5_1	90	**90** (257.354) 83 (31.647)	**90** (246.479) e6 (29.498)
RK5_2	b3	**b3** (264.236) 2c (26.498)	**b3** (226.714) bf (36.249)
RK5_3	ec	**ec** (321.698) 43 (45.268)	**ec** (314.789) f2 (33.149)

Practicality and Limitation of the attack

- IPA does not require the assumption of shared cache memory between different users. Hence, has an advantage over other types of attacks like *Prime+Probe* attacks.
- Modern processor architectures, like Intel SGX, guard against cache trace based attacks using secure enclaves. However, the event instruction count reflects the effect of cache access patterns in spite of this security.

A possible countermeasure for this attack is the implementation of block ciphers without requiring any table lookups, like NaCl [6] libraries.

Summary

- We demonstrate that surprisingly even total instruction counts can be utilized to perform side channel analysis on block ciphers.
- IPA has better performance than classical cache-timing attacks, and are better side channels than the customary timing information.
- IPA works successfully even in the presence of time obfuscating countermeasures.
- We validate our claims with results for two different environments, namely Intel and AMD.

Thank You

Robert Martin, John Demme, and Simha Sethumadhavan.
Timewarp: Rethinking timekeeping and performance monitoring mechanisms to mitigate side-channel attacks.
In *39th International Symposium on Computer Architecture (ISCA 2012), June 9-13, 2012, Portland, OR, USA*, pages 118–129, 2012.

Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3 (3A, 3B, 3C & 3D): System Programming Guide, 2010.

Debdeep Mukhopadhyay and Rajat Subhra Chakraborty.
Hardware Security: Design, Threats, and Safeguards.
Chapman & Hall/CRC, 1st edition, 2014.

Daniel J. Bernstein.
Cache-timing attacks on AES.
Techical Report, 2005.

Chester Rebeiro, Debdeep Mukhopadhyay, Junko Takahashi, and Toshinori Fukunaga.
Cache timing attacks on clefia.
In *Progress in Cryptology - INDOCRYPT 2009, 10th International Conference on Cryptology in India, New Delhi, India, December 13-16, 2009. Proceedings*, pages 104–118, 2009.

Daniel J. Bernstein, Tanja Lange, and Peter Schwabe.
The Security Impact of a New Cryptographic Library, pages 159–176.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.