An Analytical Model for Time-Driven Cache Attacks

Kris Tiri[1], Onur Acıiçmez[3]*, Michael Neve[1], and Flemming Andersen[2]

[1] Platform Validation Architecture
[2] Visual Computing Group
Intel Corporation
2111 NE 25th Avenue, Hillsboro Oregon 97124, USA
{kris.tiri,michael.neve.de.mevergnies,flemming.l.andersen}@intel.com
[3] Computer Science Lab
Samsung Information Systems America, USA
o.aciicmez@samsung.com

Abstract Cache attacks exploit side-channel information that is leaked by a microprocessor's cache. There has been a significant amount of research effort on the subject to analyze and identify cache side-channel vulnerabilities since early 2002. Experimental results support the fact that the effectiveness of a cache attack depends on the particular implementation of the cryptosystem under attack and on the cache architecture of the device this implementation is running on. Yet, the precise effect of the mutual impact between the software implementation and the cache architecture is still an unknown. In this manuscript, we explain the effect and present an analytical model for time-driven cache attacks that accurately forecasts the strength of a symmetric key cryptosystem based on 3 simple parameters: (1) the number of lookup tables; (2) the size of the lookup tables; (3) and the length of the microprocessor's cache line. The accuracy of the model has been experimentally verified on 3 different platforms with different implementations of the AES algorithm attacked by adversaries with different capabilities.

1 Introduction

Exploiting cache behavior was presumed possible by early works [7,8]. Yet, Page was the first to study the cache-based side-channel attacks. In a theoretical work [14], he classified the cache attacks based on the method of leakage observation into two types of attacks: trace-driven and time-driven cache attacks.

In trace-driven attacks (e.g. [1, 4, 9]), the adversary observes the succession of cache hits and cache misses during a cryptographic cipher operation. Given (some of) the implementation details of the cipher, she is then able to derive key material from whether a particular key-dependent memory access results in a cache hit or a cache miss. In time-driven attacks (e.g. [2, 3, 5, 16]), the adversary observes the total execution time of a cryptographic cipher operation.

* Work done while being with Intel Corporation for an internship.

She is then able to derive key material as the execution time depends on the number of key-dependent memory accesses that result in cache misses. Since the cache behavior is only one of the many elements that affect the overall execution time of a cryptosystem, time-driven attacks require statistical analysis using a large number of samples to infer key material. However, they are more generic and easier to apply than trace-driven attacks, because simple and pure software methods are sufficient to carry out time-driven attacks. On the other hand, the current trace-driven attacks are based on power analysis and mandate physical access and alteration of the processing device, and thus their use is very restricted. To the best of our knowledge, trace-driven attacks have not been demonstrated in practice.

Recently, another type of cache attack has been demonstrated: the access-driven cache attacks [11, 13, 15]. In this attack type, the adversary observes the individual cache lines accessed by the cryptographic cipher operation. She derives key material from knowing which cache lines and thus also which table entries have been touched during the key-dependent memory accesses. A single observation caries a lot of information and access-driven attacks generally require a significantly lower number of measurements than other cache attacks.

Numerous mitigations have been proposed alongside and as a reaction to the cache-attacks (e.g. [3, 5, 6, 9, 13, 14]). Most of them are software based and alter the size or the deployment of the lookup tables. For instance, with compact lookup tables [6], where a cache line contains relatively more table entries, or with dynamic lookup table permutations [6], where a cache line contains different table entries over time, less information is disclosed through knowledge of a cache line access. Yet, it is never specified how strong these mitigations are. The mitigations appear to be ad hoc and we are unaware of any formal proofs attesting their effectiveness.

Limited experimental results, however, do exist to support the mitigations [5, 6]. Yet the results are not available for all microprocessors, are only valid for a single algorithm and are unclear with respect to the actual strength of the mitigation. In this manuscript, we will present an analytical model that allows to quickly evaluate any mitigated or non-mitigated implementation on any microprocessor of any symmetric key encryption algorithm using lookup tables that are being exploited in time-driven cache attacks. The model provides a strict lower bound on the required number of measurements for a successful time-driven cache attack. The analytical model allows further to modify the threat level: it can incorporate adversaries with a sampling resolution as small as a single encryption round or as large as several complete encryptions.

The remainder of this document is organized as follows. The next section briefly describes time-driven attacks and illustrates them with the last round correlation attack. Section 3 first derives the analytical model and then shows that it is universal and valid for any time-driven cache attack as the resulting metric is based on the signal-to-noise ratio present in the measurements. In section 4, the analytical model is experimentally verified on different platforms using the unmitigated OpenSSL and several mitigated implementations of the AES algorithm. Finally a conclusion will be formulated.

2 Time-Driven Cache Attacks

The cache is a processor component that stores recently used data in a fast memory block close to the microprocessor. Whenever the processor tries to retrieve data from the main memory, it can be delivered more quickly if this data is already stored in the cache (a.k.a. cache hit). On the other hand, if the data is not available in the cache (a.k.a. cache miss), it has to be fetched from the main memory (or a higher level of cache), which has a much larger latency compared to the cache. The difference between both cache access events, i.e. a cache hit and a cache miss, is measurable and provides the attacker with information on the state and the execution of the algorithm to extract secret key material.

Observing each single cache access event, however, is extremely hard when performing a local attack and even impossible for remote attacks. Hence in a time-driven cache attack, the adversary observes the aggregated effect of all the memory accesses in a cryptographic operation, i.e. the total number of cache misses and hits or at least its effect on the execution time of the operation. To infer key material, she then analyzes cache collisions in a lookup table of interest.

In this paper, we define a cache collision as the situation involving two different memory accesses attempting to access the same memory location or different but very close memory locations that are stored in the same cache line. A cache line, aka. cache block or entry, is the smallest unit of memory that can be transferred between the main memory and the cache. The effect is ideally that the first access ensures that the data is in the cache such that the second access results in a cache hit. The attack is based on the assumption that when there is a cache collision between two particular table lookup operations, the total number of cache misses tends to be lower. On the other hand, if there is no collision between these two table lookup operations, then the overall number of cache misses tends be higher.

Note that even when there is no cache collision between the two particular table lookup operations under investigation, the second cache access might still result in a cache hit because of a cache collision with another table lookup operation. However, if enough observations are taken into account, the distribution of the number of cache misses when the cache collisions are always correctly predicted will be different from the distribution obtained when they are not. Hence, the resistance against an attack is measured as the required number of samples for a successful attack or in other words the number of samples that must be analyzed to distinguish the correct key guess from the incorrect key guesses.

An adversary tries to estimate whether a cache collision occurs between two particular table lookup operations by examining the indices of the lookup operations. She computes these indices based on the known and observable data and also based on a guess on a fragment of the secret key. Note that the secret key fragment is relatively small and that the computational complexity of a cache attack is significantly reduced compared to a brute-force attack on the entire secret key. The complete secret key is revealed by finding all of the composing key fragments.

Several statistical techniques are available to decide on the correct key value. For instance, Bonneau et al. suggest the t-test to find statistical significant different averages between distributions [5]. We use the correlation coefficient between the measurements and the estimations. This method is common practice in side-channel analysis, and especially in power analysis. This method allows us to deduce our analytical model for time-driven cache attacks that can be used to compute the strength of a given implementation on a given platform instead of relying on cumbersome empirical assessments to estimate the required number of measurements for a successful attack.

With the correlation coefficient method, the secret key fragment K_{secret} is found by evaluating the following cost function:

$$K_{secret} = max_K(|corr(M, E_K)|) \tag{1}$$

The vector M consists of the measurement scores that represent the number of cache misses that occur during the cryptographic cipher operation on the messages in a sample set. To be more precise, a measurement score is a value that approximates the number of cache misses realized during the operation such as the execution time of the operation or the cache miss count obtained via the use of performance counters. The vector E_K consists of the corresponding estimations of the adversary on the actual number of cache misses. As mentioned above, she computes these estimations by using the known values of the ciphertext or the plaintext and her guess K on a portion of the secret key.

2.1 Last round correlation attack

For a better understanding, we now describe the correlation attack on the last round of AES-128. We successfully mounted this attack on 10 different platforms (2 servers and 8 PCs) from 3 different processor manufacturers with 7 different operating systems.

Figure 1 shows a snippet of the last round of the OpenSSL AES implementation [12]. This implementation uses four lookup tables Te0, Te1, Te2, and Te3 for the first 9 rounds and a single table Te4 for the 10^{th} (i.e. last) round. Each table contains 256 4-byte words. The input to the tables is a single byte output of the preceding key addition.

The attack will estimate a single cache miss of the last round accesses and compare it with the measurement score for the total number of cache misses of the complete AES encryption. The estimation assumes that if 2 inputs to the table of interest Te4 point to the same cache line, there is a cache hit; while if they point to different cache lines, there will be a cache miss. The table indexes –i.e. `(t0>>24),(t1>>16)&0xff,(t2>>8)&0xff)`, etc. in figure 1– are a single byte of the input to the last encryption round. The model thus estimates whether $< p_i^{(10)} >$ equals $< p_j^{(10)} >$, where $p_i^{(10)}$ is the i^{th} byte input to the 10^{th} and

```
void AES_encrypt(const unsigned char *in,
                 unsigned char *out, const AES_KEY *key) {
    ...
    // apply last round and
    // map cipher state to byte array block:
    s0 =
        (Te4[(t0 >> 24)       ] & 0xff000000) ^
        (Te4[(t1 >> 16) & 0xff] & 0x00ff0000) ^
        (Te4[(t2 >>  8) & 0xff] & 0x0000ff00) ^
        (Te4[(t3      ) & 0xff] & 0x000000ff) ^
        rk[0];
    PUTU32(out     , s0);
    s1 =
        (Te4[(t1 >> 24)       ] & 0xff000000) ^
        (Te4[(t2 >> 16) & 0xff] & 0x00ff0000) ^
        (Te4[(t3 >>  8) & 0xff] & 0x0000ff00) ^
        (Te4[(t0      ) & 0xff] & 0x000000ff) ^
        rk[1];
    ...
    }
```

Figure 1. Last round snippet of OpenSSL `AES_encrypt` function [10].

last encryption round and where the $<>$ operator selects the most significant bits to account for the fact that a cache line contains several table elements ordered in function of the table index. The values $< p_i^{(10)} >$ and $< p_j^{(10)} >$ can be calculated as $< sbox^{-1}(RK_i^{(10)} \oplus C_i) >$ and $< sbox^{-1}(RK_j^{(10)} \oplus C_j) >$ based on the ciphertext bytes C_i and C_j, which are known to the attacker, and a guess on the last round key bytes $RK_i^{(10)}$ and $RK_j^{(10)}$. The correct values of $RK_i^{(10)}$ and $RK_j^{(10)}$ are the ones that result in the highest correlation coefficient between the measurements and the estimations. The key search space equals 2^{16} to find the 2 initial key bytes and $14 \cdot 2^8$ to find the remaining 14 key bytes. In the remainder of this manuscript, we will refer to this attack as the cache line estimation (CLE) attack. The attack can be simplified if the estimation neglects the fact that a cache line contains several table elements and simply assumes that only when two inputs to the table Te4 are equal, there is a cache hit; while if they are different, there will be a cache miss. In that case, the model estimates if $p_i^{(10)}$ equals $p_j^{(10)}$ or not. The substitution box is a nonlinear bijection and can be removed from the equality estimation. The model thus estimates whether $RK_i^{(10)} \oplus C_i$ equals $RK_j^{(10)} \oplus C_j$, which can be simplified further to estimating whether C_i equals $RK_{ij}^{(10)} \oplus C_j$ with $RK_{ij}^{(10)} = RK_i^{(10)} \oplus RK_j$. The correct value of $RK_{ij}^{(10)}$ is the one that results in the highest correlation coefficient between the measurements and the estimations. The key search space now equals $15 \cdot 2^8$ to find the 15 offsets $RK_{ij}^{(10)}$ from $RK_i^{(10)}$ and 2^8 to brute-force $RK_i^{(10)}$. In the remainder of this manuscript, we will refer to this attack as the table index estimation (TIE) attack. Note that the idea behind the last round correlation attack is universal and can be applied to any implementation of a symmetric key encryption algorithm using lookup tables. In the remainder of this manuscript, we will refer to the lookup table of which the collisions are analyzed as the lookup table of interest.

3 Analytical Model

For a correlation attack, Mangard derived that the number of measurements N required for a successful attack can be computed as follows [10]:

$$N = 3 + 8 \cdot \left(\frac{Z_\alpha}{\ln\left(\frac{1+\rho}{1-\rho}\right)} \right)^2 \approx \frac{2 \cdot Z_\alpha^2}{\rho^2} \tag{2}$$

The parameter ρ is the correlation coefficient between the measurement vector M and the vector $E_{K_{secret}}$, which is the estimation vector computed from the correct secret key fragment K_{secret}. The parameter α is the probability to discover the secret key, while Z_α is the quantile of the standard normal distribution for a probability α. With a probability of 0.99 to discover the secret key fragment, N can be approximated as $11/\rho^2$.

Given equation 2, the attack resistance of an implementation can be determined by (1) modeling the measurements; and by (2) computing the correlation coefficient between the estimations and the modeled measurements:

$$\rho = \frac{\mathbb{E}\left(E_{K_{secret}} \cdot M\right) - \mathbb{E}\left(E_{K_{secret}}\right) \cdot \mathbb{E}\left(M\right)}{\sqrt{\mathbb{E}\left(E_{K_{secret}}^2\right) - \mathbb{E}\left(E_{K_{secret}}\right)^2}\sqrt{\mathbb{E}\left(M^2\right) - \mathbb{E}\left(M\right)^2}} \tag{3}$$

We will construct the analytical model of the measurements based on the following 5 assumptions:

1. *The cache does not contain any data related to the cryptographic cipher operation until the operation begins.* In other words, the cache is assumed to be "clean" before the operation starts. Note that this is a valid assumption as in order for the adversary to analyze the observed side-channel information she must hypothesize a known initial state of the cache.
2. *There are no collisions between different tables used in the cryptographic cipher operation.* The cache areas on which each table maps to are mutually disjoint. Note that this is a valid assumption as the cache size of contemporary platforms is large enough to store all the tables of nearly all common symmetric key algorithms.
3. *The cache accesses during the cryptographic cipher operation are random and independent from each other.* Note that this is a valid assumption because of the avalanche effect of cryptographic algorithms.
4. *The cryptographic cipher operation operates uninterrupted.* There is no outside effect on the operation and its cache access pattern. Note that this is a valid assumption as the model models a cryptographic cipher operation. The operation can be a single encryption for a normal adversary; multiple encryptions for a limited adversary; and a single round or even a more atomic operation for a powerful adversary.

5. *The execution time of the cryptographic cipher operation is proportional to the number of cache misses.* Note that this is the foundation for time-driven cache attacks. Figure 2 shows the linear relationship between the encryption time and the total number of cache misses for the AES algorithm on an arbitrary platform. The results were computed based on 100 million encryptions of random data and averaging the encryption times of those that yield the same number of cache misses. The non-linear effects of the outliers are due to an insufficient number of those events.

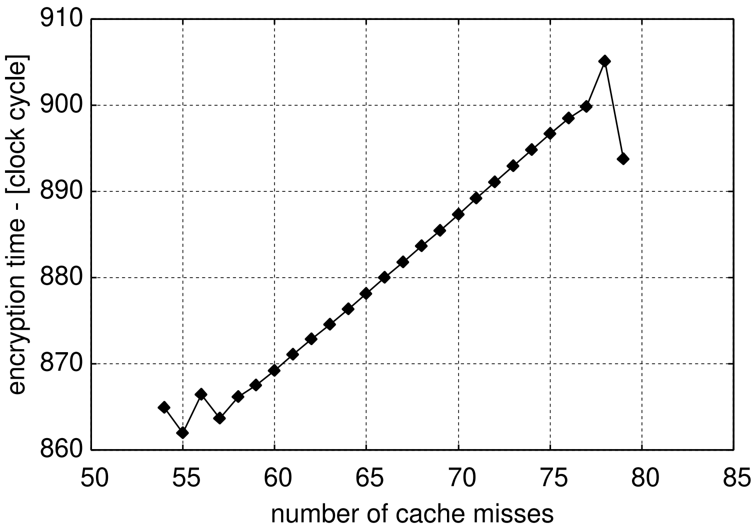

Figure 2. Linear relationship between number of cache misses and execution time.

Since the execution time is proportional to the number of cache misses, the analytical model can draw on the number of cache misses instead of on the execution time. Furthermore, using the exact number of cache misses will assure a lower bound on N since any practical measurement score for the number of cache misses will be noisier than the actual number as can be observed from the outliers in figure 2 for which insufficient samples were available to average out the noise.

We will now derive the individual components of equation 3. Since the cache accesses are independent, the 1^{st} and 2^{nd} moment of the number of cache misses equal the sum of the moments of the T tables used in the cryptographic cipher operation, as shown in equation 4. M_t denotes the number of cache misses due to accesses to table t.

$$\mathbb{E}\left(M\right) = \sum_{t=1}^{T} \mathbb{E}\left(M_t\right) \qquad ; \qquad \mathbb{E}\left(M^2\right) = \sum_{t=1}^{T} \mathbb{E}\left(M_t^2\right) \tag{4}$$

As a result, we can concentrate on calculating the moments for a single lookup table and combine them afterwards. It can be shown that the probability $P_{k,l}(j)$ that of a table occupying l cache lines exactly j cache lines are accessed after k accesses to the table is expressed by equation 5, where C_n^r: is the binomial coefficient expressing the number of combinations of r items that can be selected from a set of n items.

$$P_{k,l}(j) = \frac{C_l^j \left(\sum_{i=1}^{j} (-1)^{j-i} C_j^i i^k \right)}{l^k} \tag{5}$$

Using these probabilities, the expected value of the number of cache misses $\mathbb{E}(M)$ and the variance on the number of cache misses $\mathbb{E}(M^2) - \mathbb{E}(M)^2$ for k accesses to a table occupying l cache lines can be calculated as in equations 6 and 7 respectively. A simple expression for the expected value can be derived by noting that the probability of a single cache line not being loaded into the cache after k accesses to l cache lines is $(1-1/l)^k$. As a result, the expected number of cache lines that are not loaded becomes $l-(1-1/l)^k$ and the expected number of lines that are loaded $\mu_M(k,l)$ equals $l - l \cdot (1-1/l)^k$.

$$\mu_M(k,l) = \sum_{j=1}^{l} j \cdot P_{(k,l)}(j) = l - l \cdot \left(\frac{l-1}{l} \right)^k \tag{6}$$

$$\sigma_M^2(k,l) = \sum_{j=1}^{l} j^2 \cdot P_{(k,l)}(j) - \mu_M^2(k,l) \tag{7}$$

For ease of calculation, we now adjust the estimation to output a 1 if a cache hit has been predicted and a 0 if a cache miss has been predicted. This only changes the sign of the correlation coefficient with respect to our earlier assumption that the estimation outputs a 0 for a cache hit and a 1 for a cache miss as $\rho(A-X,Y) = -\rho(X,Y)$ with X, Y random variables and A constant.

To find the 1^{st} and 2^{nd} moment of the estimations, we note that the estimation is either a 1 or a 0. Only the 1 value will contribute to the moments, which are thus equal to $1 \cdot P(E=1)$ and $1^2 \cdot P(E=1)$, with $P(E=1)$ the probability of having an estimation equal to 1. To find $P(E=1)$, we need to differentiate between the 2 estimation models. For the table index estimation, the probability that 2 independent accesses to a table use the same table index is equal to $1/r$, with r the number of elements in the table. For the cache line estimation, the probability that 2 independent accesses use the same cache line is equal to $1/l$, with l the number of cache lines occupied by the table. The expected value of the estimated number of cache misses $\mathbb{E}(E)$ and the variance on the estimated number of cache misses $\mathbb{E}(E^2) - \mathbb{E}(E)^2$ can be calculated as in equations 8 and 9 respectively.

$$\mu_E|_{TIE} = \frac{1}{r} \quad ; \quad \mu_E|_{CLE} = \frac{1}{l} \tag{8}$$

$$\sigma_E^2 = \mu_E - \mu_E^2 \tag{9}$$

Finally, to find $\mathbb{E}(E.M)$, we need to differentiate between the table of interest and the other tables that are part of the cryptographic cipher operation. For the tables that are not of interest, the estimation and the measurements are independent and $\mathbb{E}(E \cdot M)$ is simply $\mathbb{E}(E) \cdot \mathbb{E}(M)$. For the table of interest, only the estimation with value 1 will contribute to the moment and $\mathbb{E}(E \cdot M)$ is equal to $P(E = 1) \cdot 1 \cdot \mu_H(k, l)$ where $\mu_H(k, l)$ is the expected number of cache misses with k accesses to l cache lines when a cache hit is correctly estimated. Since a cache hit will occur we can ignore the first access and the expected number of cache misses is the expected number of cache misses for the remaining $k - 1$ accesses:

$$\mu_H(k, l) = \mu_M(k - 1, l) \tag{10}$$

Using the previous equations and derivations, the correlation coefficient between the estimations and the measurement score observing the cache misses of T tables, with table T the table of interest, is equal to:

$$\rho = \frac{\mu_E \cdot \left(\sum_{t=1}^{T-1} \mu_M(k_t, l_t) + \mu_H(k_T, l_T)\right) - \mu_E \cdot \left(\sum_{t=1}^{T-1} \mu_M(k_t, l_t)\right)}{\sigma_E \sqrt{\sum_{t=1}^{T} \sigma_M^2(k_t, l_t)}} \tag{11}$$

$$= \frac{\mu_E \cdot \mu_D(k_T, l_T)}{\sigma_E \sqrt{\sum_{t=1}^{T} \sigma_M^2(k_t, l_t)}} \text{ where } \mu_D(k_T, l_T) = \mu_H(k_T, l_T) - \mu_M(k_T, l_T)$$

$$= \left(\frac{l_T - 1}{l_T}\right)^{k_r - 1}$$

Combining equation 2 and equation 11 results in the analytical model for time-driven cache attacks:

$$N = \frac{2 \cdot Z_\alpha^2}{\frac{\mu_E^2}{\sigma_E^2} \cdot \frac{\mu_D^2(k_T, l_T)}{\sum_{t=1}^{T} \sigma_M^2(k_t, l_t)}} \tag{12}$$

As an example, for the last round correlation attack using the cache line estimation to attack the OpenSSL AES implementation running on a processor with cache lines of length 64 bytes, N can be found by noting that in addition to the 16 accesses to the table of interest Te4, which occupies 16 cache lines, there are 36 accesses to each of the other 4 tables Te0, Te1, Te2 and Te3, which each also occupy 16 cache lines. With a probability for success equal to 0.99, the resulting expected number of measurements for success N is forecasted to be 6592:

$$N|_{\alpha=0.99} = \frac{11}{\frac{1/16^2}{1/16-1/16^2} \cdot \frac{\mu_D^2(16,16)}{4\cdot\sigma_M^2(36,16)+\sigma_M^2(16,16)}} = 6592 \tag{13}$$

Equation 14 tells us that for the same implementation on the same platform, the cache line estimation attack is about r/l times more effective than the table index estimation attack. This means for instance that in order to attack a regular OpenSSL implementation on a platform with 64 byte cache lines 16 times less measurements are needed when the cache miss estimation is based on cache collisions instead of internal collisions.

$$\frac{N|_{TIE}}{N|_{CLE}} = \frac{\frac{\mu_E^2}{\sigma_E^2}|_{CLE}}{\frac{\mu_E^2}{\sigma_E^2}|_{TIE}} \approx \frac{r}{l} \tag{14}$$

Equation 15 tells us that the analytical model is based on the signal-to-noise ratio that is present in the measurements and hence is independent of the actual method a time-driven attack with a given estimation model uses to select the secret key fragment. Indeed, for the same attack on two different implementations A and B on the same platform, the increase in resistance can be expressed as follows:

$$\frac{N_B}{N_A} = \frac{\frac{\mu_D^2(k_{T_A},l_{T_A})}{\sum_{t_A=1}^{T_A}\sigma_M^2(k_{t_A},l_{t_A})}}{\frac{\mu_D^2(k_{T_B},l_{T_B})}{\sum_{t_B=1}^{T_B}\sigma_M^2(k_{t_B},l_{t_B})}} = \frac{SNR_A}{SNR_B} \tag{15}$$

As can be seen in figure 3, μ_D is the distance between the distributions of the expected number of cache misses and the expected number of cache misses when a cache hit is correctly predicted. This is the signal that the adversary tries to observe and μ_D^2 is the power that is present in this signal. The noise of the measurements is the variance of the expected number of cache misses of all tables which is equal to $\sum\sigma_M^2$. The increase in resistance N_B/N_A is thus SNR_A/SNR_B, which is the ratio between the signal-to-noise ratios of the two implementations.

4 Experimental Results

The following 5 AES implementations or attack scenarios have been used to experimentally verify the correctness of the analytical model:

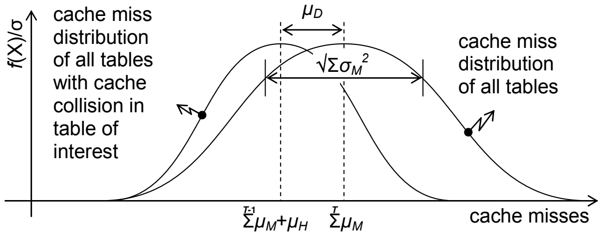

Figure 3. Cache miss probability distribution (not drawn to scale).

– OSSL (OpenSSL): The original OpenSSL implementation of the AES algorithm using 5 lookup tables Te0, Te1, Te2, Te3, and Te4, each of length 1024 bytes.
– CLR (Compact Last Round): OSSL but the last round employs a single compact S-box table of length 256 bytes instead of table Te4.
– NOT4 (NO table Te4): OSSL but the last round is implemented without the use of table Te4. Instead tables Te0, Te1, Te2, and Te3 are used.
– 2ENC (2 ENCryptions): OSSL observed by a limited adversary that is only able to observe the aggregated effect of 2 encryptions.
– OT4 (observe Only table Te4): OSSL observed by a powerful adversary who is able to observe a single round. Note that the same effect can be obtained by only cleaning out table Te4 but leaving tables Te0, Te1, Te2, and Te3 in the cache.

All experiments have been repeated on 3 different platforms from 2 different processor manufacturers with 2 different operating systems. Platform A and platform B each have L1 cache lines of length 32 bytes, while platform C has L1 cache lines of length 64 bytes. The length of the cache line is important as it specifies the number of cache lines that a lookup table occupies. If perfectly aligned in a processor with cache lines of length 32 bytes, a table of 1024 bytes occupies 32 lines, while a table of 256 bytes occupies 8 cache lines. With 64 byte cache lines, these tables occupy 16 and 4 cache lines respectively.

Since we are only interested in validating the correctness of the analytical model, the last round correlation attack of section 2.1 has been mounted in a single process setup and not with a spy and target process as would be the case for an actual attack. In the process, the cache is cleaned before the encryption of random plaintext data and the state of the performance counters, which have been programmed to count the L1 cache misses, is read just before and just after the encryption. A single measurement output consists of the ciphertext and the measurement score for the number of L1 cache misses.

Unlike the timestamp counter, the performance counters are only accessible in a high-privilege mode. Yet, a device driver and an application program interface can be installed to use the counters at any privilege level. Hence, the security evaluation of an implementation should be based on the measurement scores provided by the performance counters as this is the worst case scenario. If an adversary with access to the performance counters cannot succeed, then a more realistic adversary cannot succeed either.

Table 1 shows the required number of measurements for a successful table index estimation attack, in which the cache miss estimation is based on the table index of 2 table lookup operations. Table 2 shows the experimental results for the cache line estimation attack, in which the cache miss estimation is based on the cache line of 2 table lookup operations. The predicted results have been computed with a probability of 0.99 to discover the secret key. The measurement results are based on 100 assessments, except those that required a massive amount of measurement and processing time, which have been marked with a double dagger. For each assessment a new and random key has been generated. Note that each table entry contains 2 results: the minimum and the median required number of measurements over the 100 assessments to successfully extract the full 128-bit key. Figures 4 and 5 in the appendix are a graphic representation of the data in tables 1 and 2.

The results attest that the model accurately predicts a strict lower bound on the required number of measurements. Independent of the platform or the implementation or the adversary's observation capability, the prediction is lower than but very close to the minimum required number of measurements. Note that in order to obtain a prediction for the median required number of measurements, the confidence level, or in other words Z_α in equation 2 should be increased to account for the fact that in the majority of the experiments the correct key fragment would be extracted. Increasing the probability to discover the secret key fragment to 0.999 returns an approximate number for the median required number of measurements.

	32 byte cache line			64 byte cache line	
	predicted	measured (min/median)†		predicted	measured (min/median)†
		A	B		C
OSSL	105K	150K/230K	140K/220K	112K	160K/270K
CLR	2.06M	2.23M/3.85M	2.03M/4.00M	66.8M	109M/160M‡
NOT4	430K	420K/880K	440K/830K	1.52M	1.64M/3.30M
2ENC	234K	290K/520K	340K/510K	278K	450K/825K
OT4	12.4K	15.0K/26.0K	17.0K/26.0K	30.1K	38.0K/79.0K

†*based on 100 experiments*
‡*based on 10 experiments*

Table 1. Required number of measurements for a successful table index estimation attack.

Given that the analytical model accurately predicts the required number of measurements, it is now possible to evaluate mitigated implementations for which experimental results can not be obtained as too many measurements would be required to extract the full 128-bit key. For instance the model forecasts that changing the OpenSSL implementation to use a single compact S-box table of length 256 bytes in the first and the last round increases the required number

	32 byte cache line			64 byte cache line	
	predicted	measured (min/median)†		predicted	measured (min/median)†
		A	B		C
OSSL	12.7K	21.1K/35.6K	20.0K/32.5K	6.59K	10.0K/17.5K
CLR	56.5K	55.0K/125K	82.0K/148K	787K	1.23M/2.18M‡
2ENC	28.5K	35.0K/75.6K	33.7K/75.0K	16.3K	28.7K/49.4K
OT4	1.50K	1.87K/3.75K	2.12K/3.75K	1.77K	4.12K/7.00K

†*based on 100 experiments*
‡*based on 25 experiments*

Table 2. Required number of measurements for a successful cache line estimation attack.

of measurements to $9 \cdot 10^9$ ($\approx 2^{33}$) for an adversary that is capable to observe a single encryption on contemporary platforms with 64 byte cache lines, while using an implementation that uses a single compact table in each round increases the required number of measurements to 10^{23} ($\approx 2^{76}$).

5 Conclusions

We have provided cryptographers and software developers with a tool to evaluate the strength of their encryption algorithm or software implementation against time-driven cache attacks. The analytical model accurately forecasts the strength of a symmetric key cryptosystem based on a few simple parameters that describe the adversary's observation capabilities, the software implementation, and the platform the algorithm is running on. The accuracy of the model has been confirmed with concrete measurement results for different implementations, attack scenarios and platforms.

References

1. Onur Acıiçmez, Werner Schindler, and Çetin K. Koç. Trace Driven Cache Attack on AES. e-print of the IACR, 2006. Available online at `http://eprint.iacr.org/2006/138.pdf`.
2. Onur Acıiçmez, Werner Schindler, and Çetin K. Koç. Cache based remote timing attack on the aes. *Cryptographers' Track - RSA Conference (CT-RSA 2007)*, LNCS 4377:271–286, 2007.
3. Daniel J. Bernstein. Cache-timing attacks on AES, 2004. Available online at `http://cr.yp.to/papers.html\#cachetiming`.
4. Guido Bertoni, Vittorio Zaccaria, Luca Breveglieri, Matteo Monchiero, and Gianluca Palermo. AES Power Attack Based on Induced Cache Miss and Countermeasure. In *ITCC (1)*, pages 586–591. IEEE Computer Society, 2005.
5. Joseph Bonneau and Ilya Mironov. Cache-collision timing attacks against aes. In Louis Goubin and Mitsuru Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 2006.

6. Ernie Brickell, Gary Graunke, Michael Neve, and Jean-Pierre Seifert. Software mitigations to hedge AES against cache-based software side channel vulnerabilities. Cryptology ePrint Archive, Report 2006/052, 2006. Available online at `http://eprint.iacr.org/`.
7. John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side channel cryptanalysis of product ciphers. *Journal of Computer Security*, 8(2/3), 2000.
8. Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
9. Cédric Lauradoux. Collision attacks on processors with cache and countermeasures. In *Christopher Wolf, Stefan Lucks, Po-Wah Yau (eds.) Proceedings of Western European Workshop on Research in Cryptplogy (WeWorc 2005). GI-Edition - Lecture Notes in Informatics (LNI), P-74, Bonner Köllen Verlag (2005)*, 2005.
10. Stefan Mangard. Hardware countermeasures against dpa ? a statistical analysis of their effectiveness. In *CT-RSA*, pages 222–235, 2004.
11. Michael Neve and Jean-Pierre Seifert. Advances on access-driven cache attacks on aes. *Selected Areas of Cryptography – SAC 2006.*
12. OpenSSL. OpenSSL: the Open-source toolkit for SSL / TLS. Available online at `http://www.openssl.org/`.
13. Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache Attacks and Countermeasures: The Case of AES. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006.
14. Dan Page. Theoretical use of cache memory as a cryptanalytic side-channel. Technical Report CSTR-02-003, Department of Computer Science, University of Bristol, June 2002.
15. Colin Percival. Cache missing for fun and profit, 2005. Available online at `http://www.daemonology.net/hyperthreading-considered-harmful/`.
16. Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Maki Shigeri, and Hiroshi Miyauchi. Cryptanalysis of DES Implemented on Computers with Cache. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2779 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2003.

Appendix

Figures 4 and 5 show the required number of measurements for a successful time-driven cache attack obtained through the analytical model and obtained through experimental results for the table index estimation and the cache line estimation attack respectively. Note that these figures are a graphic representation of the data in tables 1 and 2. For completeness, the description of the attack scenarios is repeated below:

– OSSL: OpenSSL implementation of the AES algorithm.
– CLR: OSSL with compact S-box table in last round.
– NOT4: OSSL without table Te4 in last round.
– 2ENC: OSSL attacked by a limited adversary observing 2 encryptions.
– OT4: OSSL attacked by a powerful adversary observing only last round.

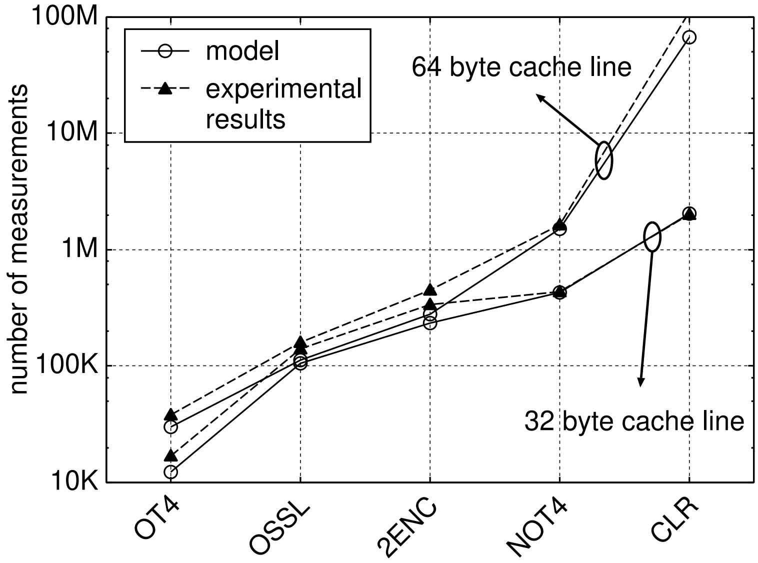

Figure 4. Required number of measurements for a successful table index estimation attack.

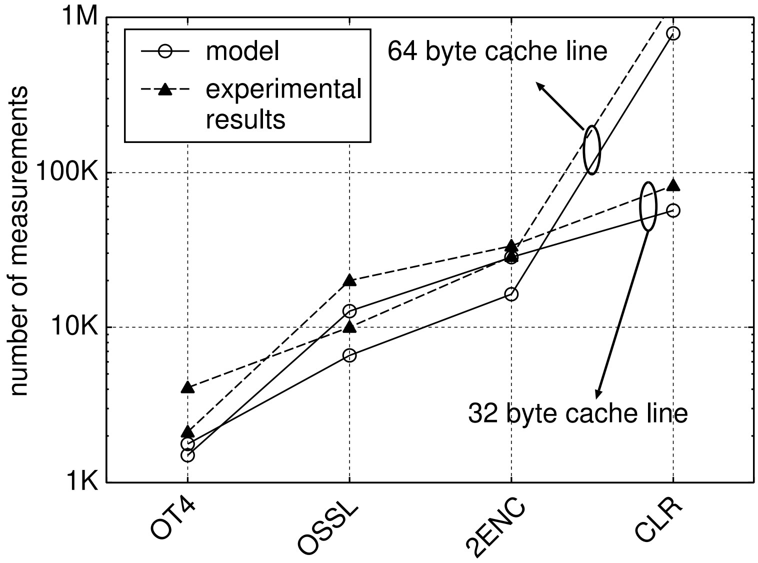

Figure 5. Required number of measurements for a successful cache line estimation attack.