Formalizing the Effect of Feistel Cipher Structures on Differential Cache Attacks

Chester Rebeiro, Phuong Ha Nguyen,
Debdeep Mukhopadhyay, and Axel Poschmann

***Abstract*—The success of a side-channel attack depends mainly on three factors, namely, the cipher algorithm, the attack platform, and the measurement noise. In this paper, we consider a class of side-channel attacks known as *differential cache attacks* on Feistel ciphers, and develop a theoretical framework to understand the relationship between the attack's success, the target platform, and the cipher algorithm. The framework allows a comparison of various differential cache attack forms, and is supported by case studies on the block ciphers CLEFIA and CAMELLIA. To understand the effect of noise in the attack's success, the paper uses empirical methods on standard Intel platforms in a time driven side-channel analysis scenario.**

***Index Terms*—Cache-timing attacks, CAMELLIA, differential cache attacks, Feistel structures CLEFIA.**

I. Introduction

Cache attacks are a potent class of side-channel attacks that utilize information leaking from memory accesses made by a cipher's implementation in order to gain knowledge of the secret key. Depending on the side-channel used, cache-attacks are categorized into three: trace-driven [1], [2], access-driven [3], [4], and time-driven [5]–[7]. Of these, time-driven attacks use variations in the encryption time to determine the secret key. They can be applied to a wide range of platforms ranging from small micro-controllers to large server blades. Further, timing measurements do not need close proximity to the device. This can lead to remote attacks [5].

The potency of cache-attacks is greatly increased when combined with classical cryptanalytic techniques. In [2], *differential cache attacks* (DCA) were introduced that used the differential properties of the s-boxes to break CLEFIA in less than 2^{14} side-channel measurements. Another form of the DCA attack on CLEFIA was presented in [8] and adopted for CAMELLIA in [9]. DCAs are especially suited for Feistel ciphers with a substitution-permutation (SP) round function, which are of growing interest in the cryptographic community with notable block ciphers (such as CAMELLIA, CLEFIA, and SMS4) adopting the structure.

The success of a DCA depends on the noise in the measurement, the cipher algorithm, and the attack platform, which in this case is the cache architecture. Changing one or more of these parameters can make a strong attack weak or vice-versa. In this paper we analyze the relationship between the success of a DCA and the factors affecting it. The contributions of the paper are as follows.

- The forms of DCAs proposed in [2] and [8] are generalized and formalized for a generic SP round function. The attacks are then theoretically analyzed in order to determine the relationship between the cipher algorithm, the cache architecture, and the number of encryptions required to determine the key. It may be noted that theoretical analysis for time-driven attacks has been previously done in [5], [10]. However to the best of our knowledge, this is the first work that shows the effect of the Feistel cipher structure on the attack's success.
- We empirically analyze the effect of noise, the attack platform, and the cipher implementation on the effectiveness of a timing side-channel.
- The effect of the various factors on the success of DCAs are presented through case studies of CLEFIA and CAMELLIA. The theoretical analysis is used to compare various DCA forms for these ciphers.

The structure of the paper is as follows: Section II has the required preliminaries while Section III discusses timing side-channels in cache-attacks. Section IV presents DCAs, and categorizes them into three categories based on the analysis method used. CLEFIA and CAMELLIA are used as case studies to evaluate the categories. The final section has the summary and conclusions.

Manuscript received November 09, 2012; revised March 20, 2013; accepted May 28, 2013. Date of publication June 11, 2013; date of current version July 03, 2013. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ramesh Karri.

C. Rebeiro and D. Mukhopadhyay are with the Department of Computer Science and Engineering, IIT Kharagpur, Kharagpur 721302, India (e-mail: chester@cse.iitkgp.ernet.in; debdeep@cse.iitkgp.ernet.in).

P. H. Nguyen and A. Poschmann are with Temasek Laboratory, NTU, Singapore 637553, Singapore (e-mail: phuongha@ntu.edu.sg; aposchmann@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2013.2267733

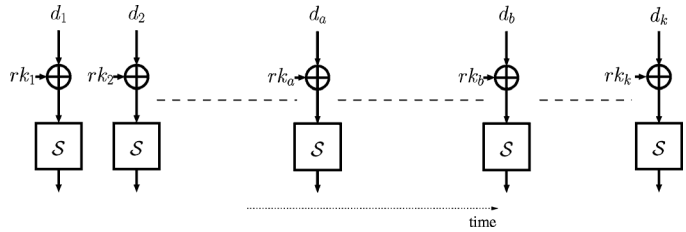

Fig. 1. Memory accesses in a block cipher implementation.

II. Preliminaries

Cache attacks target implementations of block ciphers that use lookup tables to store the s-boxes. In order to prevent the attacks, countermeasures are either applied in the application layer or in the hardware. Application layer countermeasures such as cache warming, small tables, data-oblivious memory patterns, and constant time implementations are generally bulky and do not always protect. Hardware countermeasures on the other hand have lower overheads. Many of these countermeasures propose new cache architectures. For example [11] proposes random permutations and partition locked cache architectures. Improvements were provided in [12] to prevent advanced cache attacks such as [13]. These countermeasures envisage design of new computers. Cache-attacks however remain applicable for the large number of existing systems and therefore needs to be studied. In this section we describe the basic principle of a cache attack.

Consider a 2^n element lookup table used in a block cipher implementation, which occupies 2^l memory blocks in the main memory of a processor. Let 2^v elements of the table map to the same block ($2^v = 2^{n-l}$). When any element in a block gets accessed, the entire block gets loaded into the cache memory and thereafter resides in a cache line unless evicted. From the cache-attack perspective, block ciphers can be viewed as k memory accesses made to one or more lookup tables (Fig. 1). The index of each access has the form $(d_i \oplus rk_i)$, where d_i is the data and rk_i the key material. Consider the a^{th} and b^{th} access ($1 \leq a \leq b < k$) to a table. If the accesses $d_a \oplus rk_a$ and $d_b \oplus rk_b$ fall in the same memory block, a cache hit occurs and the equality $\langle d_a \oplus rk_a \rangle = \langle d_b \oplus rk_b \rangle$ can be inferred, where $\langle \cdot \rangle$ is the most significant l bits of the table index. If d_a and d_b can be controlled

by the adversary, the ex-or of the keys rk_a and rk_b is revealed. *i.e.,* $\langle rk_a \oplus rk_b \rangle = \langle d_a \oplus d_b \rangle$.

Generally rk_a and rk_b have entropies of n bits each. If the adversary can identify a collision then the entropy of the pair of keys reduces from $2n$ to $n + v$. Identifying a collision in the memory access of the cipher can be done by monitoring side-channels of the device. The next section discusses the timing side-channel.

III. Obtaining Cache Access Patterns Using Timing

In time-driven attacks such as [5], [6], in order to observe a collision between the table accesses made by d_a and d_b (Fig. 1), the procedure presented in Algorithm 1 is followed.

Algorithm 1: Strategy in a Timing Attack

Output: The value of $\langle rk_a \oplus rk_b \rangle$ for b ($b > a$)

begin
  $d_a \leftarrow c \in [0, 2^n)$.
  for ρ number of times **do**
    $P \leftarrow (p_1 \| p_2 \| \cdots \| p_a \| \cdots \| p_m)$ where $p_a = d_a, p_i \xleftarrow{\mathcal{R}} [0, 2^n)$, and $1 \leq i \leq m$ and $i \neq a$.
    t $\leftarrow$ Encrypt using P.
    Compute $\langle d_b \rangle$ from P.
    $t_{\langle d_b \rangle} \leftarrow t_{\langle d_b \rangle} + t$; $c_{\langle d_b \rangle} \leftarrow c_{\langle d_b \rangle} + 1$
  end
  $\mathrm{t}_{avg\langle d_b \rangle} \leftarrow t_{\langle d_b \rangle} / c_{\langle d_b \rangle}$.
  $\mathrm{t}_{avg} \leftarrow$ average encryption time for all encryptions.
  return $\underset{\langle d_a \oplus d_b \rangle}{\operatorname{argmax}} (|\mathrm{t}_{avg\langle d_b \rangle} - \mathrm{t}_{avg}|)$.
end

The algorithm sets a constant value to d_a and uses it in the plaintext input P. All remaining bytes of P are chosen randomly before an encryption is triggered and timed. The time is then logged in an array indexed by $\langle d_b \rangle$. The algorithm returns the value of $\langle d_a \oplus d_b \rangle$, which has the absolute maximum deviation in timing from the average case. Algorithm 1 can be parallelized. That is for a fixed value of a, $\langle rk_a \oplus rk_b \rangle$ can be determined simultaneously for multiple values of b. In the parallel version, multiple $t_{avg\langle d_b \rangle}$ are measured (one for each b). Consequently, the algorithm would return a set of $\langle d_a \oplus d_b \rangle$. The error in detecting the collision depends on two factors : (1) the timing distributions (2) the number of measurements made (ρ in Algorithm 1). These factors are explained here.

A. The Timing Distribution

The Timing Distribution is Gaussian as seen in Fig. 2, which shows for CLEFIA, the distributions t_{avg} and $t_{avg\langle d_b \rangle}$ when there is a collision. The graph was made by executing Algorithm 1 for the first round key of CLEFIA and monitoring the time for 2^{20} encryptions on an Intel Xeon (E5606) machine. The difference between the two distributions is clearly visible.

There are two factors in the distribution that have an effect on the error in detecting the collision: the difference of means (DOM) between the two distributions and the standard deviation of the distributions. As the DOM increases it becomes easier to distinguish the distributions thus reducing the probability of error [14, ch. 4]. The DOM depends on the penalty incurred by a cache-miss and is therefore architecture specific. It can be positive or negative. A negative DOM occurs when large tables are used, in which case the difference is due to the higher number of cache misses occurring in the average case distribution. A positive

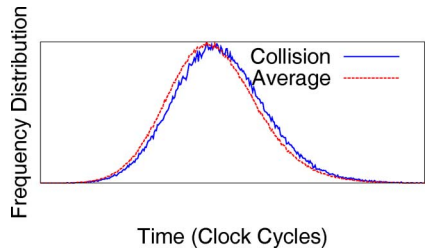

Fig. 2. Distributions of timings for CLEFIA.

TABLE I
Difference of Means and Error Probability for CLEFIA and CAMELLIA Implementations

Platform	Implement-ation	DOM (ClockCycles)	Std. Deviation (ClockCycles)	Error Probability
Intel Xeon (E5606)	CL	22.6	183	0.05
	CM	15.2	194	0.0
Intel i3 (550)	CL	9.0	396	0.0
	CM	33.8	191	0.0

CL : CLEFIA implemented with 2 tables each occupying 256 bytes
CM : CAMELLIA implemented with 4 tables each occupying 256 bytes
Xeon Cache : 32KB 8-way L1-data, 32KB 4-way L1-instruction, 256KB 8-way L2-unified, 8MB 16-way L3-unified (all have 64-byte cache line)
i3 Cache : 32KB 8-way L1-data, 32KB 4-way L1-instruction, 256KB 8-way L2-unified, 4MB 16-way L3-unified (all have 64-byte cache line)

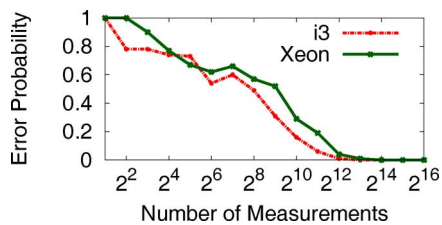

Fig. 3. Error probability versus number of measurements for CLEFIA.

DOM occurs when small tables are used. In such cases every encryption would have the entire table loaded into the cache. Here timing differences occurs due to micro-architectural effects such as out-of-order execution, parallelization, and pipelining of cache misses [15].

The standard deviations for the distributions t_{avg} and $t_{avg\langle d_b \rangle}$ are found to be almost equal. A large standard deviation results in a higher error probability. Standard deviation is affected by the noise in the measurements and is dependent on the platform characteristics such as the operating system, other processes executing concurrently, etc. The value of the standard deviation may also vary from time to time, however the DOM will be a constant for a platform. Table I shows the DOM, standard deviation, and error probability for 2 systems after 2^{20} timing measurements. The dependence of the attack platform on DOM and standard deviation can be seen. However, in all cases if sufficient measurements are made, the probability of error is close to zero.

B. The Number of Measurements

The Number of Measurements can affect the probability of error. If the timing distributions are not fully formed, then the error probability is higher. This is seen in Fig. 3, which plots error probability with increasing number of measurements (ρ in Algorithm 1) for the two implementations referred in Table I. At-least 2^{12} measurements are required to eliminate the error in detecting a collision.

IV. Differential Cache Attacks

Irrespective of the side-channel used to trace cache accesses, only the top l bits of the ex-or of the keys are obtained. In this section we

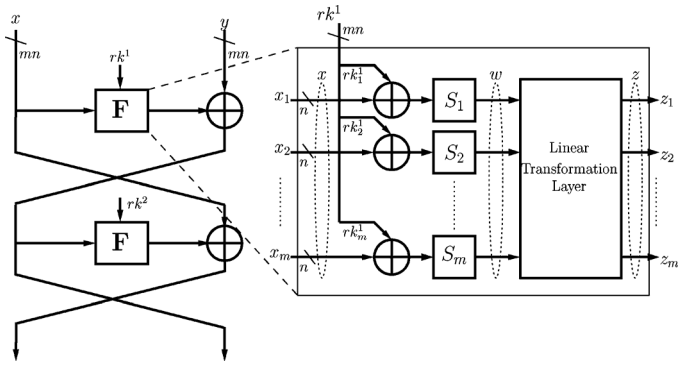

Fig. 4. Two rounds of a Feistel structure with SP-round function.

present differential cache attacks (DCA), which uses the differential properties of the cipher in addition to cache access patterns to completely obtain the secret key. The section first analyzes previous works on DCA and then presents the general technique for a new form of attack. The discussion is with respect to the Feistel structure having an SP-round function (F) as shown in Fig. 4.

Let x and y be the arms of the Feistel structure with each arm having mn bits. Each arm is split into m equal parts of n bits each (*e.g.,* $x = (x_1 \| x_2 \| \cdots \| x_m)$). The function F (in Fig. 4) can be represented as

$$z = \mathsf{P}(\mathsf{S}_1(x_1 \oplus rk_1^1), \mathsf{S}_2(x_2 \oplus rk_2^1), \ldots, \mathsf{S}_m(x_m \oplus rk_m^1)), \quad (1)$$

where S_1 to S_m are the s-boxes, P the permutation function, and rk_i^1 (for $1 \leq i \leq m$) are n bit round key parts. The s-boxes in the F function are generally implemented as lookup tables in software. The number of tables used depends on the number of different s-boxes present in the cipher. Each table would have 2^n elements and occupy 2^l memory blocks.

Suppose the cache is initially cleaned and y is chosen such that all table accesses in the second round result in collisions, the following equality is then obtained

$$\langle x_i \oplus rk_i^1 \rangle = \langle z_i \oplus y_i \oplus rk_i^2 \rangle \qquad 1 \leq i \leq m, \quad (2)$$

where z_i is the output of the F function (Fig. 4). Equation (2) can be deduced by monitoring the side channels using Algorithm 1. A cache hit in the first round can be determined by mapping $d_a \leftarrow x_i$ and $d_b \leftarrow x_j$ for $1 \leq i < j \leq m$ in the algorithm provided x_i and x_j access the same table.

A cache hit in the second round is obtained by first choosing x so that maximum number of cache hits are obtained in the first round and then y_j ($1 \leq j \leq m$) is mapped to d_b in Algorithm 1, while x_j is mapped to a corresponding d_a. Additionally x has to be kept constant. Thus (2) can be obtained with two invocations of Algorithm 1; the first invocation obtains collisions in the first round and the second invocation obtains collisions in the second round. In all 2ρ encryptions are required to be monitored.

Now consider another input $x' = (x'_1 \| x'_2 \| \cdots \| x'_m)$ such that $x' \neq x$. Let x' result in the F output z', and y' the corresponding input which causes collisions in the second round. These collisions would result in the following equalities

$$\langle x'_i \oplus k_i^1 \rangle = \langle z'_i \oplus y'_i \oplus rk_i^2 \rangle \qquad \text{where } 1 \leq i \leq m. \quad (3)$$

Adding (2) and (3) we obtain

$$\langle x_i \oplus x'_i \oplus y_i \oplus y'_i \rangle = \langle z_i \oplus z'_i \rangle. \quad (4)$$

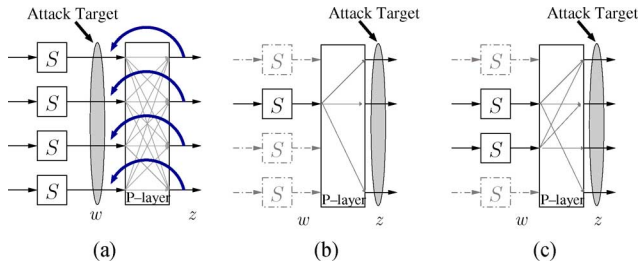

Fig. 5. Three attack schemes to derive candidates for the round key. (a) Backward attack. (b) Forward attack. (c) Generalized forward attack.

Thus the differences in the inputs $\Delta x_i = x_i \oplus x'_i$ and $\Delta y_i = y_i \oplus y'_i$ can be used to determine the output difference of the F function in the first round.

For (3), a single invocation of Algorithm 1 is sufficient because collisions in the 1^{st} round are established for free using previously determined equalities $\langle rk_i^1 \oplus rk_j^1 \rangle = \langle x_i \oplus x_j \rangle$, $1 \leq i < j \leq m$ and accesses are to the same table. Thus in total 3ρ encryptions are needed to obtain (4).

In order to derive candidates for the round key rk^1, two methods were previously devised in [2] and [8], which we call the *backward* and *forward* methods respectively (Fig. 5). We propose a *generalized forward* method and then theoretically compare all options in order to choose the best suited.

A. The Backward Attack

The backward attack was used against CLEFIA in [2] (Fig. 5(a)). It uses the inverse of the permutation layer, which can be represented as m functions having the form $w_j = LT_j^{-1}(z)$, where $1 \leq j \leq m$. For the input difference Δx, the collisions in the second round would reveal $\langle \Delta z_i \rangle$ (for all $1 \leq i \leq m$). These can be used in $LT_j^{-1}(\cdot)$ to obtain few bits of Δw_j (for all $1 \leq j \leq m$); the output difference of the s-box. The input difference and the partial output difference of the s-box is then used to reduce the key space of rk_j^1.

1) Expected Number of Candidate Keys: For any of the $n \times n$ s-boxes in Fig. 4, the total number of output differences for a given input difference is 2^{n-1}. Let o_b be the number of bits of Δw_j that gets revealed by the application of $LT_j^{-1}(\cdot)$. It can have a minimum value of 0 and a maximum of l. The possible number of output differences of the s-box gets reduced to 2^{n-o_b-1}. This reduces the set of candidate keys for rk_j^1 from 2^n to N_b. On average N_b is $2^{(n-o_b)}$.

2) Drawback: There are two potential drawbacks of this method. First, the linear transformation may not have an inverse, which would make the attack inapplicable. Second, even if the inverse did exist, the attack is useful only if the lower bits of z_i ($1 \leq i \leq m$) is not diffused into the higher bits by the function $LT_j^{-1}(z)$ (for $1 \leq j \leq m$). This is because cache accesses only reveal the high bits (*i.e.,* $\langle z_i \rangle$), therefore the diffusion of any (unknown) lower bits would mask information about Δw_j, there by preventing the attack.

B. The Forward Attack

The forward attack (Fig. 5(b)) was demonstrated on CLEFIA in [8] and CAMELLIA in [9]. Here x' is selected to be similar to x except for a difference in one part.

$$x'_i = \begin{cases} x_i & \text{if } i \neq j \\ x_i \oplus \Delta x_j & \text{if } i = j \text{ and } \Delta x_j \neq 0 \end{cases} \quad (5)$$

where $1 \leq i, j \leq m$. The small difference in x'_i becomes a random change after the s-box operation and then due to the diffusion affects at-least b s-boxes in the second round, where b depends on the branch

number of the permutation layer ($1 \leq b \leq m$). Thus y' would differ from y in at-least b parts.

The linear transformation layer can be considered as the composition of m functions, each having the form $z_i = LT_i(w)$, where $1 \leq i \leq m$. For the given input difference Δx_j ($1 \leq j \leq m$), the possible output differences of the s-box S_j is given by the set

$$W_j = \{\Delta w_j | \Delta w_j = S_j(t) \oplus S_j(t \oplus \Delta x_j), \langle LT_i(0||0||\cdots|| \Delta w_j||\cdots||0)\rangle = \langle \Delta z_i \rangle \text{ for } 1 \leq i \leq m \text{ and } 0 \leq t < 2^n\}.$$

1) Expected Number of Candidate Keys: The input change from x to x' would affect b s-boxes in the following round thus bl bits of Δz are known. This would act as an o_{f_1} *filter* reducing the number of differential outputs of the s-box from 2^{n-1} to $2^{n-o_{f_1}-1}$. The o_{f_1} filter can have values in the range from 2^l to 2^{bl} or 2^n (which ever is smaller). This results in a candidate key set for rk_j^1 (having entropy n bits) of size N_{f_1}. The average size of N_{f_1} is given by $2^{n-o_{f_1}}$.

2) Drawback: Each key part is determined individually leading to a large number of encryptions required. Another drawback of the scheme is that the information obtained about the memory accesses is not fully utilized. For example, to generate the set W_j, the fact that the first round is in a colliding state is not utilized.

C. Generalized Forward Attack

In the forward attack x and x' differed in one part ((5)). We generalize this so that x and x' can differ in more than one part. Let J be a subset of the first m positive integers. Define x' as follows:

$$x'_i = \begin{cases} x_i & \text{if } i \notin J \\ x_i \oplus \Delta x_j & \text{if } i \in J \text{ and } \Delta x_j \neq 0 \end{cases}$$

where $1 \leq i, j \leq m$. The value of x' differs from x in $|J|$ parts (where $|J|$ is the size of the set J) as seen in Fig. 5(c). Using these values of x and x', (4) is determined which would reveal parts of Δz. We now know the input difference Δx and parts of the output difference of the F function. This can be used to reduce the key space for $(rk_{j_1}^1 \parallel rk_{j_2}^1 \parallel \cdots \parallel rk_{j_{|J|}}^1)$, which has an initial entropy of $|J|n$ and $j_1, j_2, \ldots, j_{|J|} \in J$.

1) Expected Number of Candidate Keys: As in the forward attack (with $|J| = 1$), there exists an $o_{f_{|J|}}$ filter which reduces the key space of the targeted parts of rk^1 from $2^{|J|n}$ to $2^{|J|n-o_{f_{|J|}}}$. The value of $o_{f_{|J|}}$ in this case varies from $|J|l$ to $m|J|l$ (assuming that all parts of Δz are affected by the change from x to x'). The value of $o_{f_{|J|}}$ is maximum when $|J| = 1$ and reduces as $|J|$ increases. If $|J| = m$ then $o_{f_m} = m \times l$. The set of candidate keys which result from the filter has a size

$$N_{f,n|J|} = 2^{n|J|-o_{f_{|J|}}} \quad . \tag{6}$$

2) Drawback: After obtaining the input difference and output difference of the F function, the second offline phase of the attack requires to isolate all keys that satisfy this input and output difference. This requires a search of the entire key space. When $|J| = 1$, this search can be easily done as the key space of size 2^n is small. However as $|J|$ increases, the search for the key becomes difficult. In the worst case, $|J|$ is maximum and the size of the key space is as large as 2^{mn} if all the tables used in the F function are different.

D. Choosing An Attack Scheme

A *good* scheme for the DCA is one which requires minimum number of side-channel measurements (*online phase*) and also keeps the key space to be searched (*offline phase*) within practical limits. In this part of the section we provide a systematic analysis of the permutation layer and compare the most suited attack scheme for the given cache line and cipher implementation.

In all schemes except the GFA with $|J| = m$, a divide and conquer approach is adopted to obtain a part of rk^1. Let 2^t be the size of the key space targeted. For the backward attack $t = n$, since each s-box is targeted separately. While for the GFA, $t = n \times |J|$ and can be as large as mn. All attack schemes result in a set of candidate keys of size $N(< 2^t)$, which has the correct key. In order to identify which of these N keys is correct, other candidate key sets can be built with different pairs of plaintexts satisfying (4). An intersection of the candidate key sets can be applied to filter out wrong keys, since only the correct key is present in all candidate key sets. The following lemma determines the size of the intersection set.

Lemma 1: Let r be the number of candidate key sets each of size N, then the expected size of the intersection of the key sets is $N^r/2^{t(r-1)}$.

Proof: Assuming that all wrong keys are equally likely, the probability that a wrong key is present in a candidate key set is $N/2^t$. If $r = 2$, the probability that the wrong key is present in both sets is $(N/2^t)^2$, since the sets are independent. Thus the expected size of the intersection of the two sets is $2^t \cdot (N/2^t)^2 = N^2/2^t$. This satisfies the lemma.

Let the lemma hold for $r = r'$ (with $r' > 2$). Thus the size of intersection of r' sets is $N^{r'}/2^{t(r'-1)}$. Let $r = r' + 1$ and consider two sets of candidate keys. One of size N and the other the result of the intersection of r' sets. The probability that a wrong key is present in both these sets is $(N/2^t) \cdot (N^{r'}/2^{tr'})$. The expected size of the intersection is then $N^{r'+1}/2^{tr'}$, which again satisfies the lemma. ■

If r is large enough, then the intersection will uniquely identify the correct key. Let r_1 be defined as the minimum value of r to uniquely determine the correct key (assuming no key space reduction due to first round collisions for $|J| > 2$). From Lemma 1, $N^{r_1}/2^{t(r_1-1)} = 1$. Thus,

$$r_1 = \frac{t}{(t - \log_2 N)}. \tag{7}$$

The value of N depends on the diffusion layer of the F function, the number of bits revealed by the cache attack (l), and the attack scheme. From the adversary's perspective, the diffusion layer is predefined by the cipher algorithm and the cache line size is fixed by the architecture of the platform being attacked. Thus, the adversary has at disposition the choice between the forward and backward attack. For the forward attack schemes, the attack has to be repeated $m/|J|$ times in order to fully recover rk^1 (here we assume that $|J|$ divides m). Equations (6) and (7) can be used to determine the number of invocations of Algorithm 1 that are needed. Thus,

$$\text{Average Invocations of Algorithm 1 to uniquely identify } rk^1 \text{ with } GFA_{|J|} = \frac{m}{|J|} \cdot \frac{n|J|}{n|J| - \log_2 N_{f,|J|}} = \frac{mn}{o_{f_{|J|}}}. \tag{8}$$

Each invocation of Algorithm 1 would result in a set of candidate keys, which when intersected can uniquely identify rk^1. For the backward attack on the other hand, all parts of rk^1 can be targeted with the same pair of plaintexts. Thus,

$$\text{Average Invocations of Algorithm 1 to uniquely identify } rk^1 \text{ with Backward Attack Scheme} = \frac{n}{n - \log_2 N_b} = \frac{n}{o_b}. \tag{9}$$

E. Case Studies With CAMELLIA and CLEFIA

The values of o_{f_i} ($1 \leq i \leq m$) and o_b depend significantly on the diffusion layer of the F function and the cache line size. In this section we consider case studies of CAMELLIA and CLEFIA in order to present this dependence.

TABLE II
ONLINE AND **OFFLINE** ATTACK COMPLEXITY FOR A CAMELLIA IMPLEMENTATION WITH FOUR TABLES ($g = 4$), $m = 8$, AND $n = 8$

Attack Scheme	CACHE LINE SIZE							
	128 bytes		64 bytes		32 bytes		16 bytes	
	ON	OFF	ON	OFF	ON	OFF	ON	OFF
BA	8	$8 \cdot 2^8$	4	$8 \cdot 2^8$	2.6	$8 \cdot 2^8$	2	$8 \cdot 2^8$
FA	64	$8 \cdot 2^8$	32	$8 \cdot 2^8$	21.3	$8 \cdot 2^8$	16	$8 \cdot 2^8$
GFA-2	32	$4 \cdot 2^{15}$	16	$4 \cdot 2^{14}$	10.6	$4 \cdot 2^{13}$	8	$4 \cdot 2^{12}$
GFA-4	16	$2 \cdot 2^{29}$	8	$2 \cdot 2^{26}$	5.3	$2 \cdot 2^{23}$	4	$2 \cdot 2^{20}$
GFA-8	8	2^{60}	4	2^{56}	2.6	2^{52}	2	2^{48}

[1]BA is Backward Attack, GFA-i is Generalized Forward Attack with $|J| = i$, FA is the forward attack (GFA with $|J| = 1$)

1) CAMELLIA: has a classical Feistel structure with $m = 8$ and $n = 8$ (Fig. 4). The permutation layer has the property that *it only diffuses bytes, while bits within a byte are not diffused*. This means that the b^{th} bit ($0 \leq b \leq 7$) of Δw_i can only affect the b^{th} bits of Δz_j (and vice versa), where ($1 \leq i, j \leq m$). Therefore if l bits of each Δz_i are known then an equal number of bits in Δw_j can be computed. Thus the filter for the backward attack is $o_b = l$ for all possible values of j. By a similar argument the forward filter $o_{f_{|J|}} = l \times |J|$. An implementation of CAMELLIA was analyzed, which used 4 tables in the F function, each invoked twice. Table II compares the number of pairs of plaintext required (online phase) and the key space to be searched (offline phase) to uniquely identify rk^1 for different attack schemes and cache line sizes. The table was obtained by estimations using (8) and (9). As an example, for the backward attack with cache line size of 128 bytes, 8 pairs of plaintext satisfying (4) are required to uniquely identify rk^1. To find these 16 plaintexts (8 pairs) requires 16 invocations of Algorithm 1. Each pair of plaintext produces a set of candidate keys having an offline search space of 8×2^8 (8 because there are 8 bytes in the key). The property that bits within a byte are not diffused makes the backward attack the best scheme. The only other scheme to have an equivalent online complexity is the GFA with $|J| = 8$. However, there is a huge offline search space of 2^{60}.

2) CLEFIA: uses two F functions named $F0$ and $F1$ each having $m = 4$ and $n = 8$ (see Fig. 4). Diffusion is provided by MDS matrices, which operate in $GF(2^8)$, are self inverting and unlike CAMELLIA, *diffuse bits within a byte*. Due to the latter property, the backward attack will not work for certain cache lines. To see this consider for example the byte w_1 in the $F1$ function. Each bit of w_1 can be computed from z (Fig. 4) using inverse of the diffusion layer functions ($LT_i^{-1}(\cdot)$ for $0 \leq i \leq 7$) as follows: $w_{1,0} = LT_0^{-1}(z_{2,7}, z_{4,7}, z_{3,5}, z_{4,5}, z_{1,0})$, $w_{1,1} = LT_1^{-1}(z_{3,6}, z_{4,6}, z_{1,1}, z_{2,0}, z_{4,0})$, $w_{1,2} = LT_2^{-1}(z_{2,7}, z_{3,7}, z_{3,5}, z_{4,5}, z_{1,2}, z_{2,1}, z_{4,1})$, $w_{1,3} = LT_3^{-1}(z_{2,7}, \ldots, z_{3,5}, z_{4,5}, z_{1,3}, z_{2,2}, z_{4,2}, z_{4,0}, z_{3,0})$, $w_{1,4} = LT_4^{-1}(z_{2,7}, \ldots, z_{3,5}, z_{4,5}, z_{1,4}, z_{2,3}, z_{4,3}, z_{3,1}, z_{4,1})$, $w_{1,5} = LT_5^{-1}(z_{3,7}, z_{4,7}, z_{3,6}, z_{4,6}, z_{1,5}, z_{2,4}, z_{4,4}, z_{3,2}, z_{4,2})$, $w_{1,6} = LT_6^{-1}(z_{3,7}, z_{4,7}, z_{1,6}, z_{2,5}, z_{4,5}, z_{3,3}, z_{4,3})$, $w_{1,7} = LT_7^{-1}(z_{1,7}, z_{2,6}, z_{4,6}, z_{3,4}, z_{4,4})$, where $z_{i,j}$, is the j^{th} bit of x_i, where $0 \leq i \leq 4$ and $0 \leq j \leq 7$.

If $l < 4$, the cache traces only reveals bits of $z_{i,b}$ for $4 < b \leq 7$ then no bits of w_1 can be computed. Therefore l has to be at-least 4 for the backward attack to work (in which case $o_b = 1$). Table III shows the relationship between o_b and l for the permutation layers in $F0$ and $F1$. The table was obtained by estimations from (8).

For the forward attack o_f can be larger than l thus achieving a stronger filter. For example consider the case of attacking rk_j^1 in $F1$ with $|J| = 1$ (where $1 \leq j \leq 4$). The diffusion of $LT_i(0\|\cdots\|\Delta w_j\|\cdots\|0)$ results in $\langle 1 \cdot \Delta w_j \rangle$, $\langle 2 \cdot \Delta w_j \rangle$, $\langle 8 \cdot \Delta w_j \rangle$, $\langle a \cdot \Delta w_j \rangle$. This allows more than l bits to be compared, as seen in Table III (*i.e.,* $o_{f_1} > l$). However the GFA with $|J| = 1$ allows only one byte of key to be targeted at a time, therefore 4 times more pairs of plaintext are required to completely recover rk^1. This results in a trade off between a strong filter and the number of pairs of plaintext required. This trade off makes the backward attack more suited in some cache lines and forward attack more suited in others as seen in Table IV. The table shows the attack complexity for the online and offline phases of a CLEFIA implementation which uses two 256 byte tables each invoked twice per F function. It can seen that on large cache lines the backward scheme is not feasible (marked NA in the table) while no such limitation is present for the GFA, where all cache line sizes can be attacked. Even where the backward attack works, there are certain cache lines where the forward attack performs better. This is because in these cache lines the o_{f_i} ($1 \leq i \leq m$) filter is strong enough to overcome the number of pairs of plaintexts required due to the sequential nature of the attack. For example consider the cache line size is 16 bytes. The filter $o_b = 2$ and $o_b = 1$ for $F0$ and $F1$ respectively (Table III). The attack recovers the entire round key with 4 and 8 pairs of plaintexts (Table IV). For the same cache line size, the GFA filters (for $|J| = 1$) are comparatively much stronger. However, the GFA being sequential, can only target one byte at a time. Therefore the attack needs to be repeated 4 times in order to recover the entire round key. This deteriorates the effectiveness. For $F1$ the filter ($o_{f_1} = 7$) is strong enough so that the attack performs better than the backward scheme as seen in Table IV. However, this is not so for $F0$ ($o_{f_1} = 6$), where the backward scheme is better.

TABLE III
FILTER o_b AND o_f FOR CLEFIA IN BITS FOR DIFFERENT CACHE LINE SIZES

Filter	Cache Line Size (in bytes)							
	$F0$				$F1$			
	128	64	32	16	128	64	32	16
BA (o_b)	NA	NA	1	2	NA	NA	NA	1
FA (o_{f_1})	3	4	5	6	3	5	6	7
GFA-2 (o_{f_2})	4	8	10	12	4	8	12	14
GFA-4 (o_{f_4})	4	8	12	16	4	8	12	16

TABLE IV
ONLINE AND **OFFLINE** ATTACK COMPLEXITY ON CLEFIA IMPLEMENTED WITH TWO TABLES ($g = 2$), $m = 4$, AND $n = 4$

Attack Scheme	CACHE LINE SIZE							
	128 bytes		64 bytes		32 bytes		16 bytes	
$F0$	ON	OFF	ON	OFF	ON	OFF	ON	OFF
BA	NA	NA	NA	NA	1	$4 \cdot 2^8$	2	$4 \cdot 2^8$
GFA-1	3	$4 \cdot 2^8$	4	$4 \cdot 2^8$	5	$4 \cdot 2^8$	6	$4 \cdot 2^8$
GFA-2	4	$2 \cdot 2^{15}$	8	$2 \cdot 2^{14}$	10	$2 \cdot 2^{13}$	12	$2 \cdot 2^{12}$
GFA-4	4	2^{30}	8	2^{29}	12	2^{28}	16	2^{27}
$F1$	ON	OFF	ON	OFF	ON	OFF	ON	OFF
BA	NA	NA	NA	NA	NA	NA	1	$4 \cdot 2^8$
GFA-1	3	$4 \cdot 2^8$	5	$4 \cdot 2^8$	6	$4 \cdot 2^8$	7	$4 \cdot 2^8$
GFA-2	4	$2 \cdot 2^{15}$	8	$2 \cdot 2^{14}$	12	$2 \cdot 2^{13}$	14	$2 \cdot 2^{12}$
GFA-4	4	2^{30}	8	2^{29}	12	2^{28}	16	2^{27}

F. Correctness of the Estimations

Algorithm 1 forms the basis for executing a DCA. The number of invocations of Algorithm 1 required to perform a DCA depends on the permutation layer of the cipher, the attack scheme, and the cache line size. This has been theoretically formulated in Section IV-D. To validate the theory we executed Algorithm 1 for the block ciphers CLEFIA and CAMELLIA by simulating caches and collisions in P-functions, and then determined the average number of invocations of the algorithm that were required to uniquely identify the round key rk^1 in each case for different cache line sizes. Fig. 6 compares simulation results with the results obtained from the theory for a few of these configurations. There is a small difference between the theoretical and simulated results because the s-boxes in the ciphers are not truly random; an assumption behind Lemma 1. Nevertheless a close match between theory and practice can be observed.

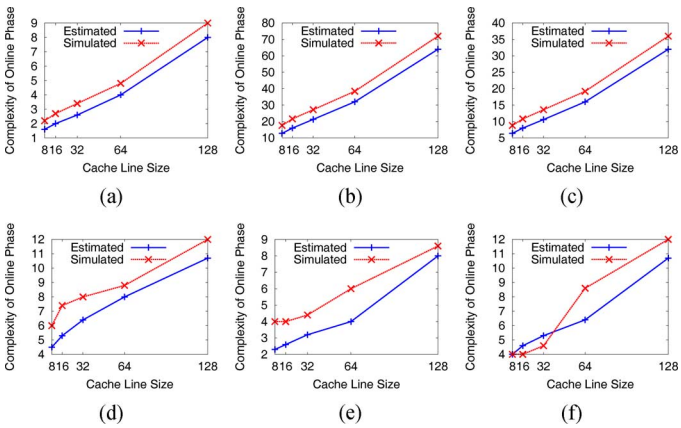

Fig. 6. Validation of estimated complexity of the online phase with simulated results. (a) CAMELLIA BA. (b) CAMELLIA FA. (c) CAMELLIA GFA-2. (d) CLEFIA F0 FA. (e) CLEFIA F0 GFA-2. (f) CLEFIA F1 FA.

V. Summary and Conclusions

The paper analyzes the various factors that affect the success of a DCA. A framework is developed to assess the vulnerability due to the cipher algorithm, cache architecture, and attack schemes. The permutation layer in the cipher algorithm along with the cache line size are the most influential factors that affect the attack's success. For permutation layers that only diffuses bytes and not bits within a byte (*e.g.,* CAMELLIA), the backward scheme is preferred because it can be parallelized and has a filter which is as strong as that of the forward scheme. For ciphers that diffuse bytes as well as bits within a byte (*e.g.,* CLEFIA), the best attack scheme depends on the number of key parts that are targeted. If the filter's strength dominates the number of key parts then the forward scheme is more effective than the backward scheme. However if the number of key parts dominates the filter's strength then the backward scheme is more suited. The paper also empirically analyzes noise in timing side-channels. Results show that the cipher implementation and attack platform are important factors that affect the success with which a collision is detected. However, if sufficient number of measurements are made, then the collision present during the encryption is always detected.

References

[1] G. Bertoni, V. Zaccaria, L. Breveglieri, M. Monchiero, and G. Palermo, "AES power attack based on induced cache miss and countermeasure," in *Proc. ITCC*, 2005, vol. 1, pp. 586–591, IEEE Computer Society.

[2] C. Rebeiro and D. Mukhopadhyay, "Cryptanalysis of CLEFIA using differential methods with cache trace patterns," in *Proc. CT-RSA*, A. Kiayias, Ed., 2011, vol. 6558, Lecture Notes in Computer Science, pp. 89–103, Springer.

[3] M. Neve and J.-P. Seifert, "Advances on access-driven cache attacks on AES," in *Selected Areas in Cryptography*, E. Biham and A. M. Youssef, Eds. New York, NY, USA: Springer, 2006, vol. 4356, Lecture Notes in Computer Science, pp. 147–162.

[4] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: The case of AES," in *Proc. CT-RSA*, D. Pointcheval, Ed., 2006, vol. 3860, Lecture Notes in Computer Science, pp. 1–20, Springer.

[5] O. Aciiçmez, W. Schindler, and Ç. K. Koç, "Cache based remote timing attack on the AES," in *Proc. CT-RSA*, M. Abe, Ed., 2007, vol. 4377, Lecture Notes in Computer Science, pp. 271–286, Springer.

[6] Y. Tsunoo, T. Saito, T. Suzaki, M. Shigeri, and H. Miyauchi, "Cryptanalysis of DES implemented on computers with cache," in *Proc. CHES*, C. D. Walter, Ç. K. Koç, and C. Paar, Eds., 2003, vol. 2779, Lecture Notes in Computer Science, pp. 62–76, Springer.

[7] J. Bonneau and I. Mironov, "Cache-collision timing attacks against AES," in *Proc. CHES*, L. Goubin and M. Matsui, Eds., 2006, vol. 4249, Lecture Notes in Computer Science, pp. 201–215, Springer.

[8] C. Rebeiro, R. Poddar, A. Datta, and D. Mukhopadhyay, "An enhanced differential cache attack on CLEFIA for large cache lines," in *Proc. INDOCRYPT*, D. J. Bernstein and S. Chatterjee, Eds., 2011, vol. 7107, Lecture Notes in Computer Science, pp. 58–75, Springer.

[9] R. Poddar, A. Datta, and C. Rebeiro, "A cache trace attack on CAMELLIA," in *Proc. InfoSecHiComNet*, M. Joye, D. Mukhopadhyay, and M. Tunstall, Eds., 2011, vol. 7011, Lecture Notes in Computer Science, pp. 144–156, Springer.

[10] K. Tiri, O. Aciiçmez, M. Neve, and F. Andersen, "An analytical model for time-driven cache attacks," in *Proc. FSE*, A. Biryukov, Ed., 2007, vol. 4593, Lecture Notes in Computer Science, pp. 399–413, Springer.

[11] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *Proc. ISCA*, D. M. Tullsen and B. Calder, Eds., 2007, pp. 494–505, ACM.

[12] J. Kong, O. Aciiçmez, J.-P. Seifert, and H. Zhou, "Hardware-software integrated approaches to defend against software cache-based side channel attacks," in *Proc. HPCA*, 2009, pp. 393–404, IEEE Computer Society.

[13] J. Kong, O. Aciiçmez, J.-P. Seifert, and H. Zhou, "Deconstructing new cache designs for thwarting software cache-based side channel attacks," in *Proc. CSAW*, T. Jaeger, Ed., 2008, pp. 25–34, ACM.

[14] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. New York, NY, USA: Springer, 2007.

[15] C. Rebeiro, D. Mukhopadhyay, J. Takahashi, and T. Fukunaga, "Cache timing attacks on CLEFIA," in *Proc. INDOCRYPT*, B. Roy and N. Sendrier, Eds., 2009, vol. 5922, Lecture Notes in Computer Science, pp. 104–118, Springer.