# Differential Analysis of a Low Energy Table-based Countermeasure for Secure Embedded Systems

C.Gebotys,
Dept of E&CE
University of Waterloo
cgebotys@uwaterloo.ca

**ABSTRACT**

**Future wireless embedded devices will be increasingly powerful supporting many more applications including one of the most crucial, security. Although many embedded devices offer more resistance to bus probing attacks due to their compact size, susceptibility to power or electromagnetic analysis attacks must be analyzed. This paper presents a table masking countermeasure to resist differential power analysis (DPA) and differential electromagnetic analysis (DEMA). Real power and EM measurements are used to verify the countermeasure using 2nd and 3rd order DPA and DEMA attacks on a popular low energy embedded ARM processor. Results show that the new table masking countermeasure provides increased security without large overheads of energy dissipation compared to previous countermeasures. With the emergence of security applications in PDAs, cellphones, and other embedded devices, low energy countermeasures for resistance to DPA/DEMA is crucial for supporting future wireless embedded systems.**

## 1. Introduction and Related Research

As more security applications migrate to the wireless device, resistance to attacks on the PDA or cellphone will become a necessity. These attacks may not only arise from device theft or loss but also during everyday use where unintentional electromagnetic (EM) waves radiated from the wireless device during cryptographic computations may leak confidential data to a nearby attacker. For example an attack may be successful in obtaining the secret keys stored in confidential memory in a wireless device. This attack may be possible through loss or theft of the device, or alternatively through temporary access to the device by monitoring the EM waves emanating from the device while performing cryptographic computations. In the latter case the attack may be able to extract the encryption keys (typically not even accessible to the user of the PDA), making all wireless communications insecure. Researchers have already demonstrated that this new attack is viable[7,10,29] on a 8-bit processor. Since EM waves are highly correlated with power, ensuring wireless devices are secure from power analysis attacks is important. Nevertheless large overheads in energy to achieve this resistance may not be practical. Unfortunately cryptographic algorithms are already known to consume significant amounts of energy [2]. Even worse, cryptographic algorithms which are resistant to attacks are known to have latency overheads up to 1.9 times[5]. Outside of smartcard research (which typically has been in the past limited to cheaper 8-bit or 16-bit processors) [6,7], few researchers have examined secure implementations of cryptographic software under the threat of power attacks on 32-bit processors. For example a very popular embedded processor core is the ARM which is suitable for portable devices

ranging from game devices to PDAs to cryptographic applications [5,16,17]. There is an important need to study energy optimized countermeasures for processor cores which will be incorporated into SoCs in wireless portable devices, such as PDAs, cellphones, etc.

Currently research in power attacks of smart cards, have utilized 8-bit general purpose processors [5,3,12] and 5V processors running at 4MHz[4,1]with an average current consumption of 10mA[3]. Typically smart card applications are not time critical and energy dissipation is not a major concern since power is attained from the card reader (or ATM machine, etc). The measurement of power while a processor is executing an application (or a power trace) has been used in power-attacks of cryptographic devices, such as smart cards[1]. In particular the analysis of the variation of power, and computations on a number of power traces can be used to detect data and algorithmic dependencies[1]. Previous research studied the correlation of power variation with data values being manipulated and instruction sequencing. In the former case, known as differential power analysis (DPA), encryption[1] and public key encryption[22] applications were used[1]. Differential power attacks of embedded low power processors have not been reported in the literature, apart from frequency based extensions of DEMA[33,34] and low power DSP processor power analysis[36]. Higher order differential attacks [15,35] are an extension of the $1^{st}$ order DPA which involve using joint statistics on multiple points within power traces.

Random sequencing of instructions (desynchronization), randomized power noise (through hardware circuits) and balancing methods[11,13] are some suggested countermeasures against power analysis. In the later technique computations are performed not only on the data but also on the complement of the data (where words contain for each of the data bits, it's complement [11]). Although researchers have suggested the balancing method may fail[13] due to the difference in the two complementary values, few researchers have used real power measurements in conjunction with DPA to verify this claim. Other researchers designed bus complement techniques in conjunction with bus precharging to resist DPA attacks [7]. Only a few researchers [7] have demonstrated this with simulation techniques, however none have verified this countermeasure using real power measurements. Other techniques which combine desynchronization and randomization have also been researched such as clock gating to protect against DPA [6]. However the impact of power randomization on DPA has also not been published with real power measurements. Additionally research [32] has examined VLSI techniques for resistance to power analysis.

In addition to the above countermeasures, other approaches have been suggested such as secret splitting[9], duplication method[18], multiplicative masking[19] and random masking[5]. Secret splitting, involves splitting the secret data into smaller pieces and combining them with random data [9]. Then the cryptographic algorithm is run on each word, which is composed of random and secret data. To attack the splitting method, a $k^{th}$ order differential attack is required[9] (where secret data is split into $k$ shares), however no statistics were derived to be used in this attack. The duplication method[18] was used to support secure computations with multiple split variables for input to the cryptographic tables or S-boxes ($S[x]$). These researchers also used table duplication such that one table contained a randomly-chosen secret transformation on $x$, $A[x]$, and the alternate table contained $A[x] \oplus S[x]$. Multiplicative masking was also defeated by a DPA attack[19]. In the random masking countermeasure, each secret piece of data is exclusive-or'd with a random data value (called a mask). To thwart a DPA attack the random data value must be changed periodically. However this involves remasking the tables (or exclusive-or the complete table data with a mask) within the algorithm. Countermeasures have been reported to run up to 1.9 times slower [5] for Rijndael (or Rijndael, an advanced encryption standard [27]). These overheads are largely due to remasking of tables, so some researchers have investigated storing a limited number of masked tables [14] (referred to as the fixed value masking approach). The

authors define the S-box DPA as an attack on the (masked or unmasked) output of the S-box table (or output of the table look-up in figure 1a)). It is assumed that the attacker has control over the input plaintexts which are exclusive-or'd with the key to index the S-box table (or the attacker can observe the ciphertext for a similar attack). Using probabilistic DPA the authors[14] proved that the fixed-value masking approach was secure as long as the masks were chosen appropriately (the probability of the $i^{th}$ bit of a fixed mask being equal to zero is ½). However results using real power measurements were not performed to verify the fixed value masking approach. A second order DPA attack was developed [15], for example in an attack on the input of the S-box, to thwart the random masking countermeasure (also known as "data whitening"). A heuristic for the $2^{nd}$ order DPA statistic was developed and applied with real power measurements to a 8-bit processor at 3.57MHz using a sample rate of 1G samples per second. The $2^{nd}$ order DPA converged slowly for some bits requiring approximately 2500 power traces. A $2^{nd}$ order DEMA attack was launched in [10] on a 8-bit processor with 500 EM traces and it was shown to produce better results than the $2^{nd}$ order DPA. Attacks using power samples of data and correlating them with the hamming weight of the data were studied in [23] for Rijndael, however masking was not considered.

Countermeasures must be energy optimized for wireless device implementation. Furthermore if these countermeasures can be defeated then they must require high order differential analysis (DEMA) in that a large number of power/EM traces must be required making the attack very difficult. The use of real EM measurements of low power embedded 32-bit processors is important to verify these countermeasures for portable applications and in SoCs where power may not be readily accessible. This paper presents a low energy countermeasure and differential power and EM analysis results for a popular low power ARM embedded processor. The numbers of power and EM traces required in the attacks on a 32-bit processor are computed for both real power and real EM measurements. The next section will provide an introduction to DPA/DEMA followed by the proposed table masking countermeasure and the application of the countermeasure to Rijndael.

### 1.1 A Brief Introduction to Differential Power Analysis

A brief introduction to the differential analysis attack is presented in this section. Power traces are used to describe the attack however it is also applicable to electromagnetic traces (or the demodulated EM traces as described in [10]).

Consider figure 1a) which illustrates a common part of many cryptographic algorithms, especially when executed on 32-bit processors (where table look-ups provide a much faster implementation of bit level computations). Typically the input data or plaintext and key are involved in some computation whose result is indexed into a table look-up (or S-box table). Assume for now that the plaintext and key are exclusive-or'd together and then indexed into a table, as in the table method of the Rijndael advanced encryption standard[28] or fast implementation of DES. The attacker has control over the plaintext and knows the values in the table (according to some well published encryption algorithm). The attacker wishes to determine the key value.

First the current drawn from the device (or processor) is recorded while the device is executing the cryptographic algorithm. The following variables will be used to describe the subsequent processing: $C_i(t)$ represents the instantaneous power signal at time $t$ of the $i^{th}$ power trace, where $i=1,\ldots,n$; and $j$ is the bit of the data output from the S-box table in the $i^{th}$ execution of the algorithm. The attacker makes a guess at the key value, for example a guess for the 8-bits of the key ( {???? ????} in figure 1b)), whose value is exclusive-or'd with the known plaintext (a: xxxx xxxx or b: xxxx xxxx in figure 1b)) and input

to the S-box table. The attacker next partitions the power traces into two groups according to expected value of bit *j* (a zero, *0?*, or one, *1?*) of the S-box table output. One group contains all power traces where the value of bit *j*, or $b_j$, of the S-box table output is expected to be *0* ( a: *x…0?…x* ) and the other group contains all power traces whose value of bit *j* of the S-box table output is expected to be *1* ( b: *x…1?…x* ).

Let $x_1(t) = \{C_i(t) \mid \forall i = 1,..n1, b_j = 0\}$ and $x_2(t) = \{C_i(t) \mid \forall i = 1,..n2, b_j = 1\}$ be the two groups of power traces. The mean of each group of power traces is computed (to average out the impact of the other non-*j* bits). The difference of means or differential power trace is obtained by subtracting the two averaged power traces ( *{a:x…0?…x }– {b:x…1?…x}* ) . This subtraction attempts to remove the algorithmic impact on the power trace thus leaving only the $j^{th}$ bit effect on the power trace. The differential power trace is equal to :

$$\overline{x_1(t)} - \overline{x_2(t)}$$

This difference of means is significant if it is much greater than the standard deviation of the difference of means. In this paper we define the DPA characteristic for the guessed key as the maximum value of the absolute differential power trace minus 2 standard deviations (of the difference of means) at that time. In other words the DPA characteristic (for each key guess) is equal to:

$$\max\left[\left|\overline{x_1(t)} - \overline{x_2(t)}\right| - 2\sqrt{\frac{\sigma_1(t)^2}{n_1} + \frac{\sigma_2(t)^2}{n_2}}\right]$$

where $\sigma_k$ is the standard deviation of group *k* and $n_k$ is the number of samples in group *k* (where *k*=1 or *k*=2). This attack is repeated for all possible values of the key (256 for an 8-bit key) and the maximum DPA characteristic is used to determine which key guess is correct.

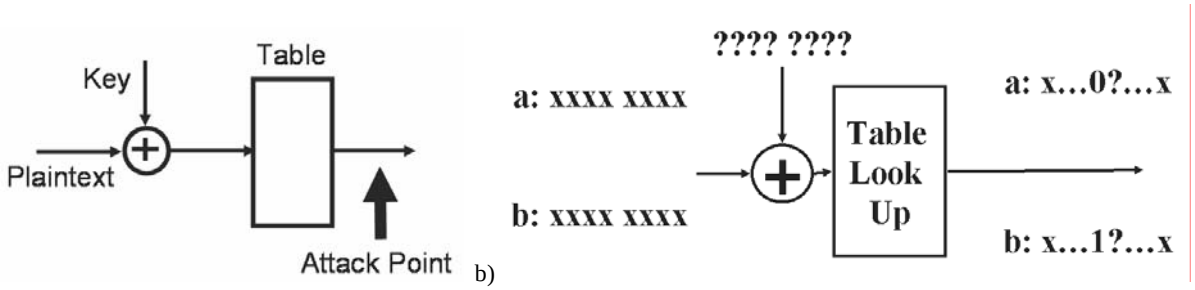



**Fig. 1. Attack point for S-box DPA/DEMA in a) and partition of traces in b)**

The next section will present the table masking countermeasure for the S-box DPA/DEMA in a general example. The detailed implementation of the table masking countermeasure for Rijndael is then presented in the following section along with a security analysis of the countermeasure. Finally experimental results are presented using real power and EM measurements for a popular ARM 32-bit processor.

## 2. Table masking Countermeasure for DEMA

The table masking countermeasure stores randomly masked S-box data in a masked S-box table, *S'[x]*. Unlike previous research [5,14], each addressed data in the table uses a different random mask (*S'[x] = S[x]* $\oplus$ *r[x]* , *r[x]*= random data which is different for each table address, *x*). A second

corresponding table (called the mask table, $M[x]$) is used to store a corresponding mask for each address ($M[x]=r[x] \oplus m$), such that the exclusive-or of the masked S-box table and the mask table is a fixed masked value ($m= r[x] \oplus M[x]$, for all $x$ ). However the final mask, $m$, is always split into two values, $r[x]$ and $M[x]$, and is never computed. The algorithm for computing the masked S-box table and mask table, shown in figure 2 (where the small circle with a + in the middle represents an exclusive-or operation), is performed before the cryptographic algorithm is downloaded to the device, hence an attacker does not have access to $m$ (nor can the attacker obtain the hamming weight of $m$ from power measurements). Since tables are generated only once, this value $m$ along with the mask of the round keys is used to precompute tables before the cryptographic algorithm is downloaded to the device. To avoid a first order DPA, the exclusive-or of the S-box masked table and mask table, $S[x] \oplus r[x] \oplus M[x]=m \oplus S[x]$ , is never computed during the execution of the encryption algorithm. Figure 3 illustrates the computations performed on the data accessed in Rijndael from the mask tables before they are merged with the data accessed from the masked S-box tables.

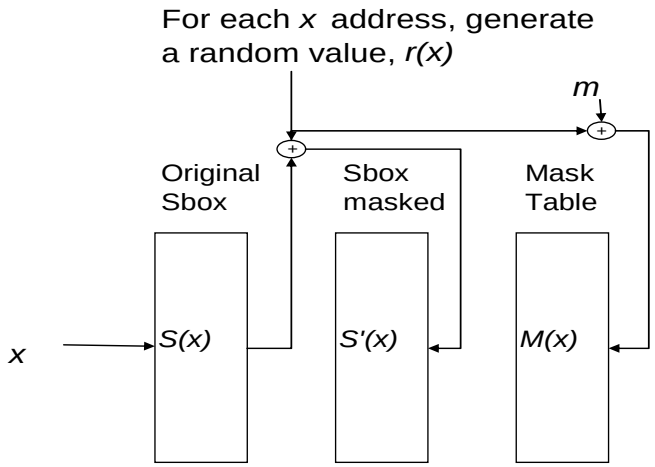



**Fig. 2. Generation of S-box Masked Table and Mask Table**

For example, in the table method of Rijndael (described in [28] for fast implementation on 32-bit processors), all data accessed from the masked S-box tables are exclusive-or'd together and then exclusive-or'd with the round key (as would normally be done for unmasked S-box tables in the original algorithm) to produce a value *t0a*. Next all corresponding data accessed from the mask table are exclusive-or'd together to produce a value *t0b*. Finally these two values are exclusive-or'd together. For example in the table method of Rijndael one would normally compute :

$t0 = S1(\{s0\}_{b3}) \oplus S2(\{s1\}_{b2}) \oplus S3(\{s2\}_{b1}) \oplus S4(\{s3\}_{b0}) \oplus rk_i$

(where $\{w\}_b$ refers to byte $b$ of the 32-bit word $w$). However with this countermeasure one would compute :

$t0a = S1'(\{s0\}_{b3}) \oplus S2'(\{s1\}_{b2}) \oplus S3'(\{s2\}_{b1}) \oplus S4'(\{s3\}_{b0}) \oplus rk_i$

$t0b = M(\{s0\}_{b3}) \oplus M(\{s1\}_{b2}) \oplus M(\{s2\}_{b1}) \oplus M(\{s3\}_{b0})$

$t0 = t0a \oplus t0b$

(where *t0* value is input to the next set of tables, which may also have masked inputs). Figure 3 illustrates the computations required in the scheme for Rijndael, where *S1', S2', S3', S4'* and *M* are the four masked S-box tables and the one mask table, respectively. Note that there is only one mask table,

M, which is accessed four times (so it is shown four times) in figure 3. In this scheme if tables are generated with one value of *m*, as illustrated in figure 2, then *t0* will be unmasked (since the four *m's* cancel). In order to produce a masked value of *t0*, at least one or all *S'j[x]* tables can be defined as $S'j[x]=Sj[x] \oplus r[x] \oplus mj$, where *mj* is a random mask for table *j*. In this approach the final mask on *t0* would be $m1 \oplus m2 \oplus m3 \oplus m4$ in Rijndael since there are 4 S-box tables (*j=1…4).*

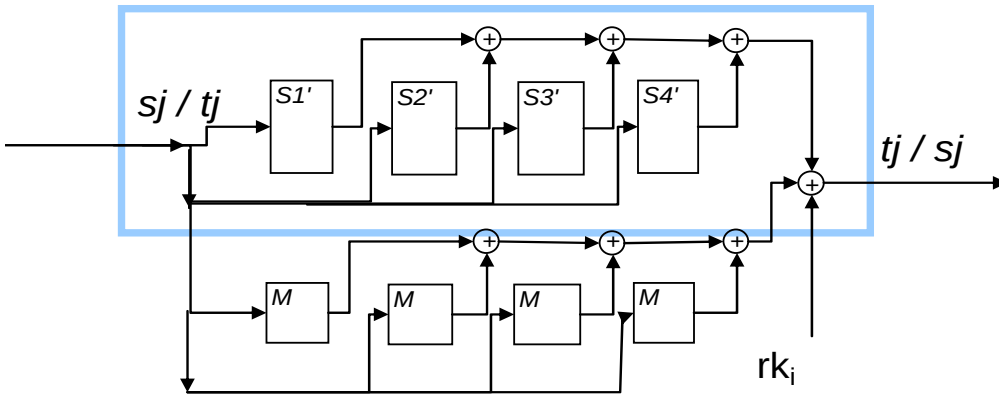



**Fig. 3. Partial Rijndael implementation of proposed Countermeasure with only one mask table(M) used for all S-box masked tables (S1',S2',S3',S4').**

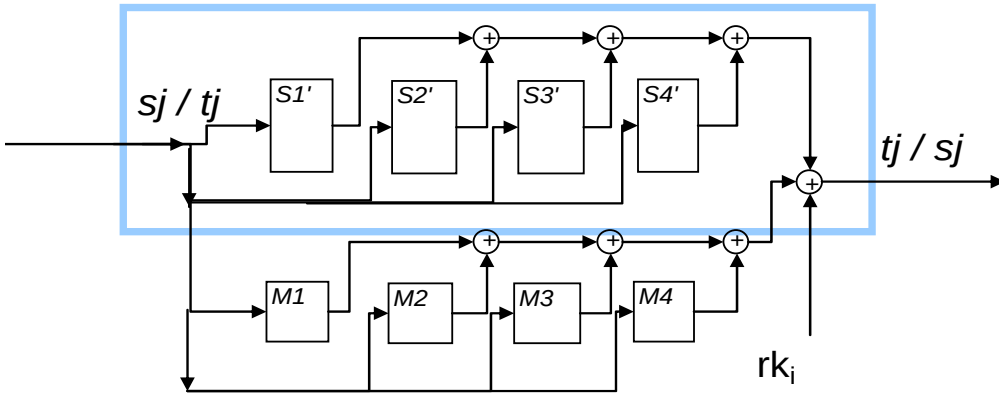



**Fig. 4. Partial Rijndael implementation of proposed Countermeasure with a separate mask table (M1,M2,M3,M4) for each S-box masked table (S1',S2',S3',S4').**

If the set of *r[x]* is the same set used for all S-boxes then only one *M[x]* table needs to be stored, as shown in figure 3. However if storage is not an issue in the embedded system, a different set of *r[x]* for each S-box could be used, to increase the number of different random masks. For example a separate mask table, *Mi[x]*, for each S-box, *Si'[x]*, could be implemented as shown in figure 4. Furthermore dynamic updating of random masks *ri[x]* is easily supported in this scheme. For example after access of data from both *Mi [x]* and *Si'[x]*, a remasking of these two data values can be computed by exclusive-oring them with another new random value. Then these newly masked values can be stored back into the original *Mi[x]* and *Si'[x]* tables. This procedure for dynamically updating or refreshing random masks (during the en/de-cryption) is shown in figure 6.

A similar implementation for DES and other cryptographic algorithms which use look up tables is also possible. The implementation of the table masking for DES is shown in figure 5. The computations in the left hand column represent normal DES with no countermeasure implemented. The rectangular oval shapes represent the expansion-permutations in DES. The masked S-boxes (*S'_1,2,3,4,5,6,7,8*) provide both S-box substitutions and P-box permutations. The rectangular boxes are just used to illustrate input and output data of each round (where round 1 uses round key $k_1$, round r uses round key $k_r$ and round 16 uses round key $k_{16}$, shown on the left). In DES there are 8 S-box tables, hence in this countermeasure 8 masked S-box tables (*S'_1,2,3,4,5,6,7,8*) and one extra mask table, *M*, are used. Initially for round 1, it is assumed that the round key $k_1$ is masked (where the mask is equivalent to the input table mask of the S-box masked tables, which is our fixed mask *m*). Subsequent rounds repeat the operations shown in round r.

## 3. Security Analysis of Countermeasure

A 1st order S-box DEMA/DPA is thwarted since in this table masking countermeasure, the data values output from the S-box tables have been decorrelated through random masking. For example bit *i* of *r[x]* values which partly define the masked S-box and mask tables should have an equal probability of being a *0* or *1* for different values of *x*. The randomization can also be modified by dynamically refreshing random masks as previously described in section 2 and in figure 6.

Similar to previously researched countermeasures[5,18], a 2nd order differential analysis [15,30] on this implementation of the countermeasure may be possible. However unlike previous countermeasures [5,18], a 2nd order differential analysis can be thwarted by using more than one mask table (for example using $M_1[x]$ and $M_2[x]$, in place of *M[x]*, and $m = r[x] \oplus M_1[x] \oplus M_2[x]$ ). For example in figure 7, *n* mask tables can be created by further splitting the masks. Hence the security of this countermeasure scales with the number of tables. For *n* mask tables ($M_1, M_2 \ldots, M_n$ ), and one set of masked S-box tables, *(S'* ), a $(n)^{th}$ order DEMA attack is thwarted (where $m = r[x] \oplus M_1[x] \oplus M_2[x] \ldots \oplus M_n[x]$, for all *x* , thus splitting mask *m* into *n+1* masks ). For example in Rijndael, a 2nd order DEMA can be thwarted with a storage overhead of 2 extra mask tables ($M_1$, $M_2$ where $m=r[x] \oplus M_1[x] \oplus M_2[x]$, for all *x*). The higher order attack typically will require many more EM traces and thus provides an increase in difficulty of launching the attack. Note that once again, even with *n* tables, to avoid a nth order DEMA, the exclusive-or of the S-box masked table and mask table, specifically $S'[x] \oplus M_1[x] \oplus M_2[x] \ldots \oplus M_n[x] = m \oplus S[x]$ is never computed.

The proposed table masking countermeasure is similar to the duplication method [18] however there are two important differences. First, unlike [18], the second table does not hold the random mask of the S-box entry, *r[x]*. Second the duplication method [18] used two tables whereas this countermeasure can increase the number of tables thwarting higher order DEMA. In the former case, the proposed mask table holds the value $m \oplus r[x]$, not *r[x]* as in [18]. Given this difference the 2nd order attack is different than the one presented in [15]. Appendix A presents two attack statistics, one where *m* is known and the other for *m* unknown. Furthermore if *m* changes for different iterations of the encryption algorithm, then the statistic presented in Appendix A does not work. In fact it is not clear how to formulate a statistic for this case, given that each trace may have been generated with a different value of *m*. The value of *m* can be changed after a large number of algorithm executions (ie. a number less than the number of traces required for a successful 2nd order DEMA), by creating a random number, *r*, and exclusive-oring the mask table with this value ($M[x] \oplus = r$ *, for all x).*

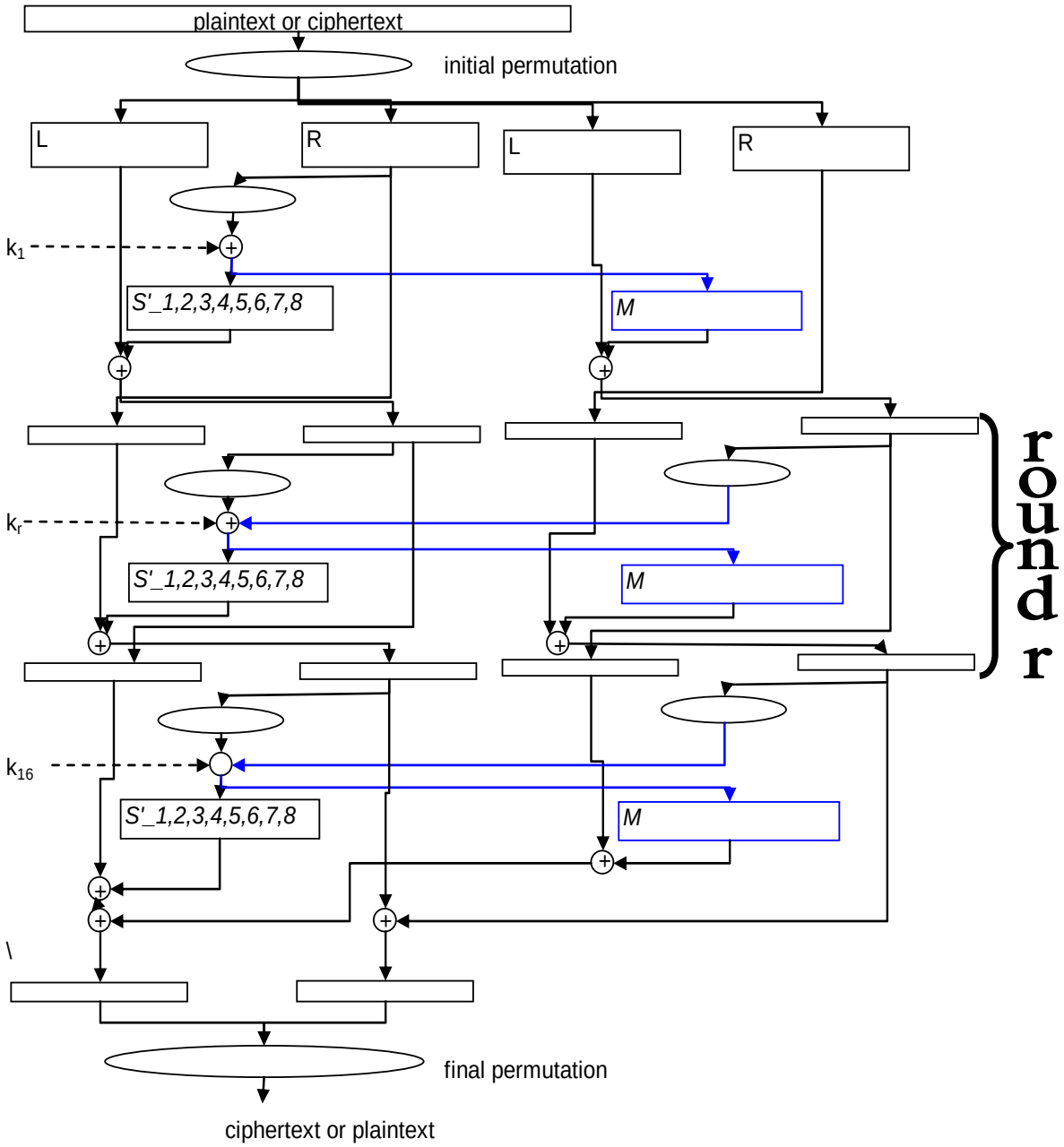



**Fig. 5. Countermeasure implemented with DES where right hand column defines computations with mask table.**

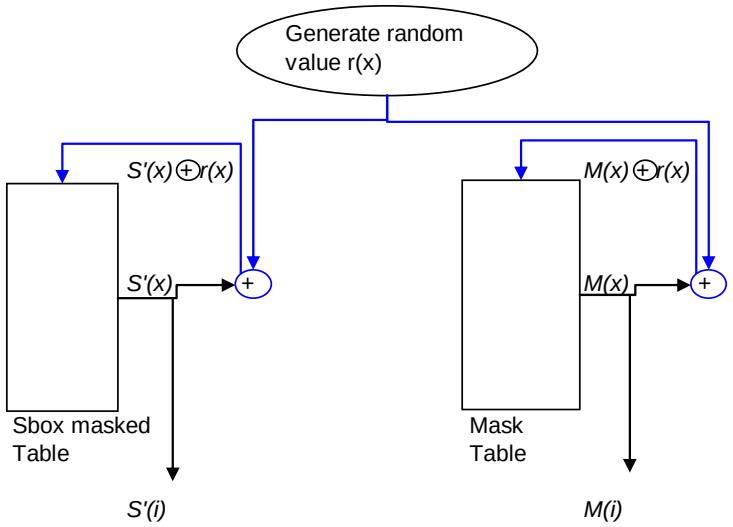



**Fig. 6. Dynamic refreshing of random masking on an address in one (all) S-box masked table(s) and corresponding mask table.**

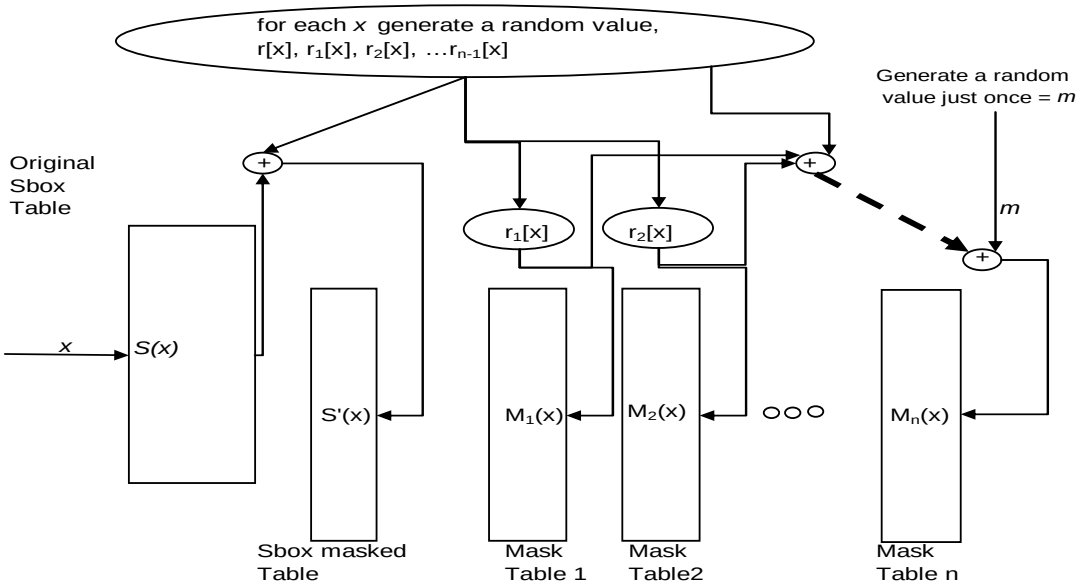



**Fig. 7. Generation of n mask tables for one/all S-box tables, in order to make the attack harder.**

If an attacker has access to the masked tables (for example by dumping the memory contents of the device) and *m* does not change, it may be possible to derive the fixed values in the masked S-boxes and then perform a 1st order attack to obtain the key (assuming the key is not stored in the accessible memory). For example if the attacker exclusive-or's the masked S-box table with the mask table(s), the resultant value, $M[x] \oplus S'[x] = m \oplus S[x]$, is still masked with *m*, unlike [18] where access to the tables immediately reveals the location of the S-box values and random values through exclusive-oring (where $M[x] \oplus S'[x] = S[x]$). The attacker next must exclusive-or two values together, $(m \oplus S[x]) \oplus (m \oplus S[y]) = S[x] \oplus S[y]$, for two values of *y*, and compare these with precomputed pairwise exclusive-ors generated by the attacker with the known S-box values. When both matches occur (ie. $S[x] \oplus S[y]$ *and* $S[x] \oplus S[z]$),

the value of *S[x]* is determined in the *S'[x]* masked S-box table. With this knowledge, the value of *m* can be determined and finally used to locate the remaining S-box values in the masked S-box. Given the location and value of masked S-box values, a 1st order DEMA can be launched to obtain the secret key. However if *m* changes, as described in the previous paragraph, then even this attack is thwarted.

## 4. Experimental Setup

A high sample rate oscilloscope, a trigger probe, an EM probe, an inductive probe and an ARM7TDMI evaluation board (providing access to the core's power supply) were used to acquire power and EM traces. The inductive probe measured the instantaneous current drawn on the 3.3V ARM7TDMI processor core power supply line. In this paper we will refer to the processor current as the power consumed (since the supply voltage was assumed to be stable at 3.3V). The EM probe (consisting of a 1cm loop passive magnetic field probe) was placed on top of the ARM7TDMI packaged chip, see figure 8b). A broadband preamplifier was also used to boost the signal to noise ratio of the EM signals. The trigger signal was controlled by software and measured with a probe in order to synchronize the measurements. The experimental setup is illustrated in figure 8a). The scope sampled at rates up to 2.5Gsamples/sec, and allowed many power and EM traces to be acquired automatically. The evaluation board, figure 8b), had the 16/32-bit ARM7TDMI RISC processor core on one chip separate from the memory. Hence all the power measurements reflect the processor core power consumption only and not the input/output buffer power or memory power. The ARM7TDMI could be set to different clock frequencies (up to 56MHz) and utilized a three stage pipeline. This ARM processor core is often referred to as a low power processor consuming on average 0.6mA/MHz at 3V. Figure 8a) illustrates the detailed connections and figure 8b) provides a photo of the board with the EM probe and inductive probe for EM and power measurements.

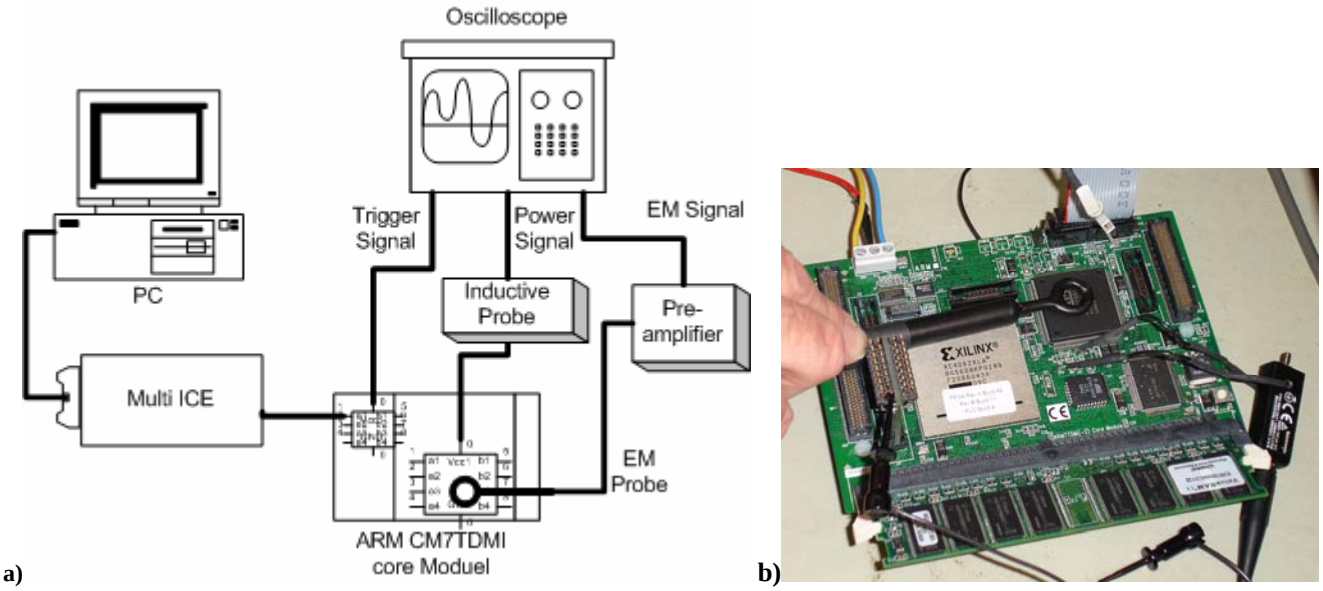



**Fig. 8 Experimental setup for measuring the EM emissions and power drawn from the processor with photo of board.**

Each scope run acquired up to 6000 power or EM traces (referred to as a frameset) and acquisitions were repeated three times to ensure high reliability and reproducibility in the results. The ARM7TDMI processor was programmed to use a 40MHz clock for all experiments. All results were

repeatable and where possible the same set of plaintexts (or input data to the cryptographic algorithm) was used to compare results of different DPAs/DEMAs and countermeasures. All DPAs/DEMAs were verified as significant using two standard deviations.

To acquire the multiple samples in each power/EM trace for the 2nd and 3rd order DEMA experiments, the cryptographic programs were first characterized. Specifically 1st order DEMAs were run to identify the best power sample points within the specific load and stores within the cryptographic program. The highest point of the positive DEMA characteristic was used to identify the sample point in the EM trace which would be used as the EM samples ($b1_k$, $b2_k$, $c_k$ in Appendix A,B) in the high order DEMAs. Each frameset of EM traces acquired from the scope was characterized wth DEMAs to determine the best sample point. Those sample points were then fixed for all experiments utilizing that cryptographic program. The next section will present the DEMA/DPA and statistic results utilizing real EM/power measurements obtained with the ARM7TDMI and high sampling rate scope. Unless otherwise stated and unlike research in [10], no demodulation of the EM signals were performed.

## 5. Experimental Results

This section will illustrate results of the proposed table masking countermeasure and high order DPA/DEMA attacks using both real power and real EM measurements. Rijndael and DES were used to illustrate the countermeasure and higher order analyses. The attack point of Rijndael, as previously discussed, was at the output of the S-boxes, whereas in DES, the attack point was after the exclusive-or which followed the S-box access. Experimental results using both a commercial and hand-made probe to acquire EM signals, with and without demodulation, are also reported. In all plots where the x-axis denotes the number of traces, the numbers reflect the number of traces in one of two partitions in the differential analysis. Hence the total number of acquired traces in these experiments is actually double the number shown on the x-axis.

Figure 9a) presents the plot of DEMA height (solid line) versus number of EM traces. The dashed line indicates two times the standard deviation of the difference of means (indicating significance when the DEMA height exceeds two standard deviations). Two example plots are shown in figure 9a) and 9b). In both cases bit 1, the least significant bit, was used to partition EM traces and produce the first order DEMAs. Two different acquisitions with the same set of data were used to generate the two figures, hence the EM noise is the main factor in the difference in the EM analysis results. In all cases the two standard deviation drops off as the number of traces increases and the DEMA height levels off. It is interesting to note that in figure 9a) only 100 EM traces (in each partition, totaling 200 EM traces) are required for a significant DEMA peak, whereas in figure 9b) over 1000 EM traces (in each partition) are required for the same data, acquired at different times. The DEMAs, for bit 1 through bit 4, are shown in figure 9c) and 9d) which correspond to the data acquired in Figure 9a) and b) respectively. The faster converging data has stronger DEMAs as evident by the blue DEMA exceeding the red two standard deviations in figure 9c). However in figure 9d) the DEMAs are not as strong hence supporting the slower convergence time shown in figure 9b) for the same set of data.

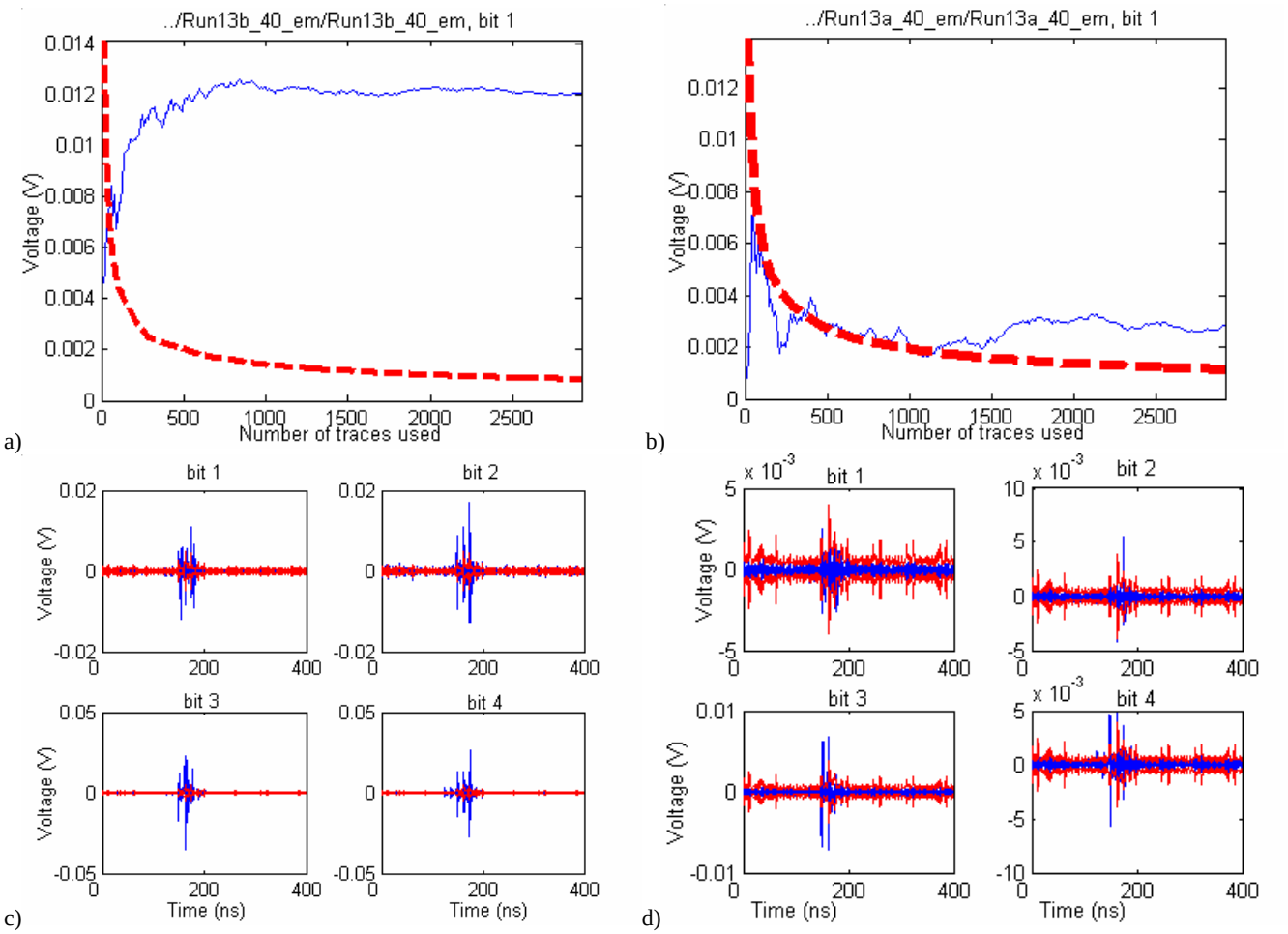


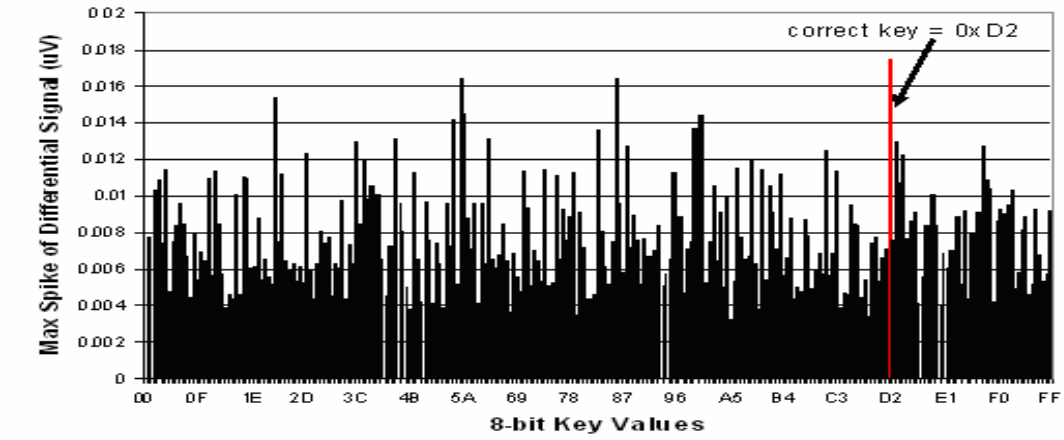

**Fig. 9 Convergence of 1st order DEMA using bit 1 for two EM acquisitions of same data showing two standard deviations (dashed) and height of DEMA characteristic (blue) in a),b) and in c),d) corresponding DEMAs for bit 1,2,3,4 using same two data acquisitions respectively.**

Figure 10 shows the result of the 1st order DEMA at the output of the S-box in Rijndael using the ARM7TDMI processor in the evaluation board. The scope captured 2976 different EM traces (each obtained using different plaintexts as input to the Rijndael algorithm). The raw EM signals were then used to obtain the difference of means as described in section 1.1 for each guess of the secret key. In figure 10a), the resultant DEMA characteristic (y-axis) for each key guess (x-axis) is shown. The largest DEMA characteristic illustrates the expected correct key. The experiment illustrated by the bar chart will be called the all keys guess analysis. Figure 10b) illustrates the same attack as in figure 10a), except the Rijndael code now has the proposed table masking countermeasure implemented. The correct key is identified by the arrow and is the lighter colored bar. Clearly the correct key is not evident (ie. the highest value) in figure 10b), hence the countermeasure has successfully resisted the DEMA attack.

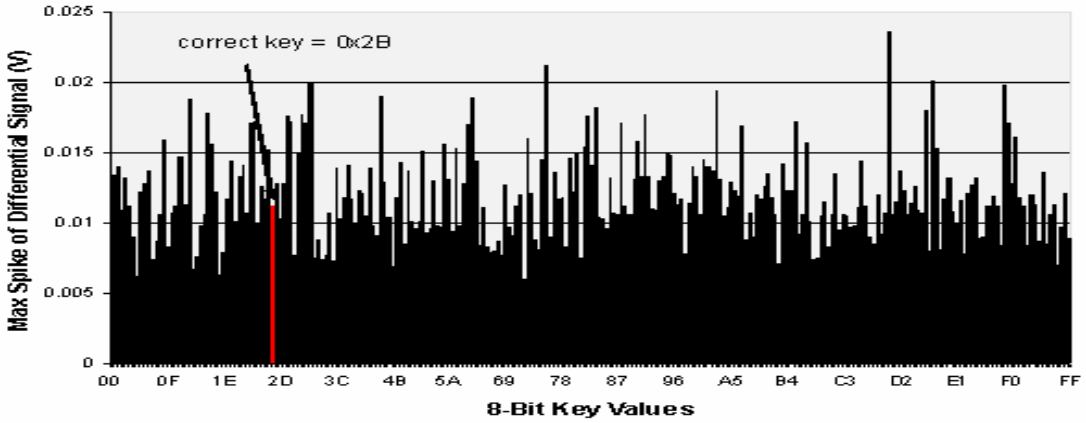
In figure 11, a hand-made probe (denoted as *white probe*), b), is compared to the commercial probe in a) for Rijndael. The effect of demodulation on these EM traces was further explored in figure 11c) and d) for the commercial and hand-made probe respectively. Specifically the EM signals were demodulated with a bandwidth of 50MHz at the ninth harmonic of the clock (or 360MHz). The hand-made probe performed just as well as the commercial probe, however had slightly lower amplitudes.

The demodulation did not work as well as the raw EM signals in the differential analyses. All the experiments were repeatable.

The results for a first order DPA and DEMA on DES are shown in figure 12a) and 13a) respectively for power and EM traces. Figures 12b) and 13b) illustrate the effect of the proposed countermeasure which thwarts the 1st order DEMA respectively. The correct key is shown as the darker vertical line in these figures.

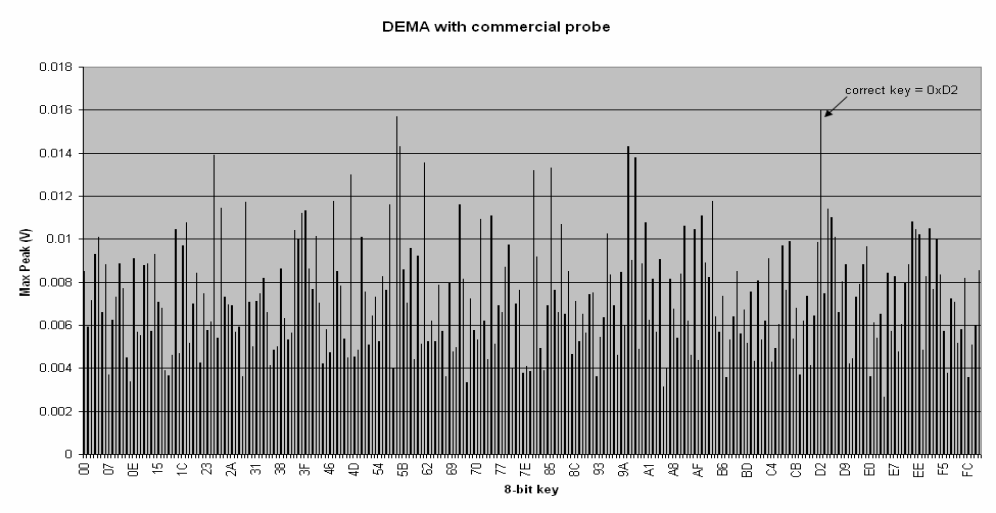



a)

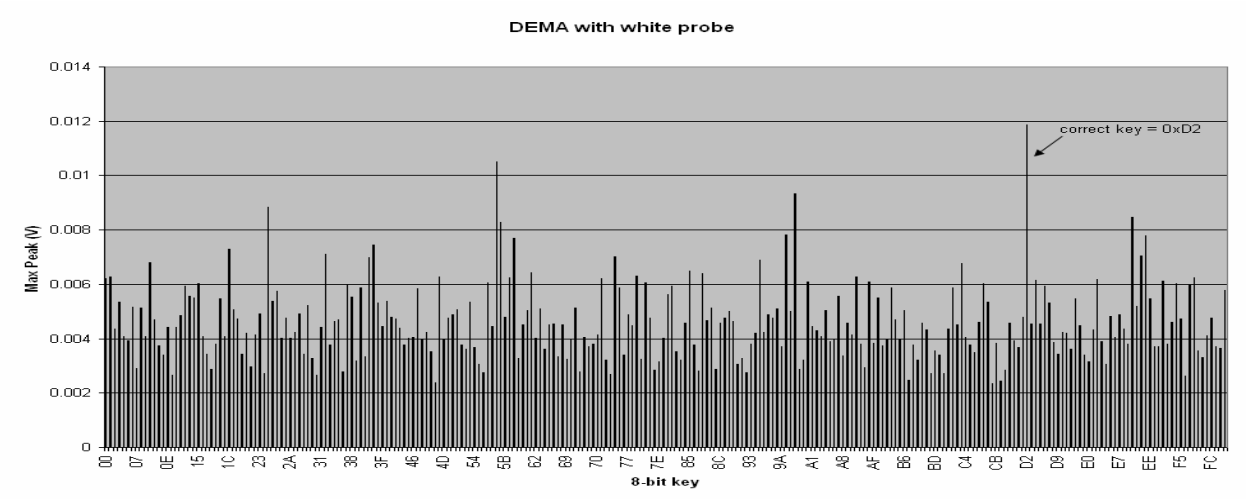



b)

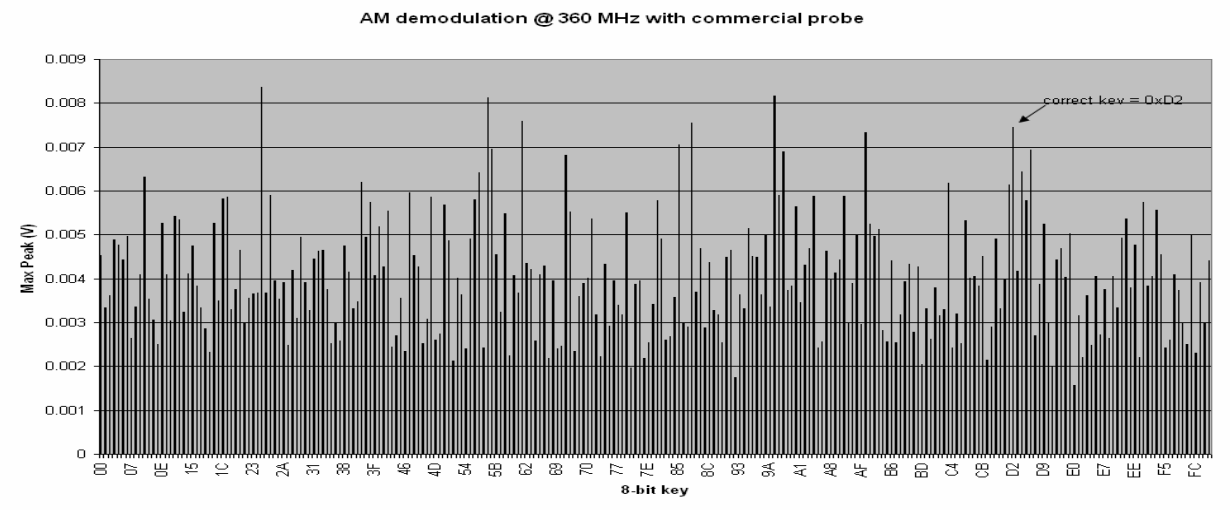
**Fig. 10. 1st order DEMA on Rijndael without and with countermeasure in a) and b) respectively.**


DEMA with commercial probe
correct key = 0xD2
Max Peak (V)
8-bit key

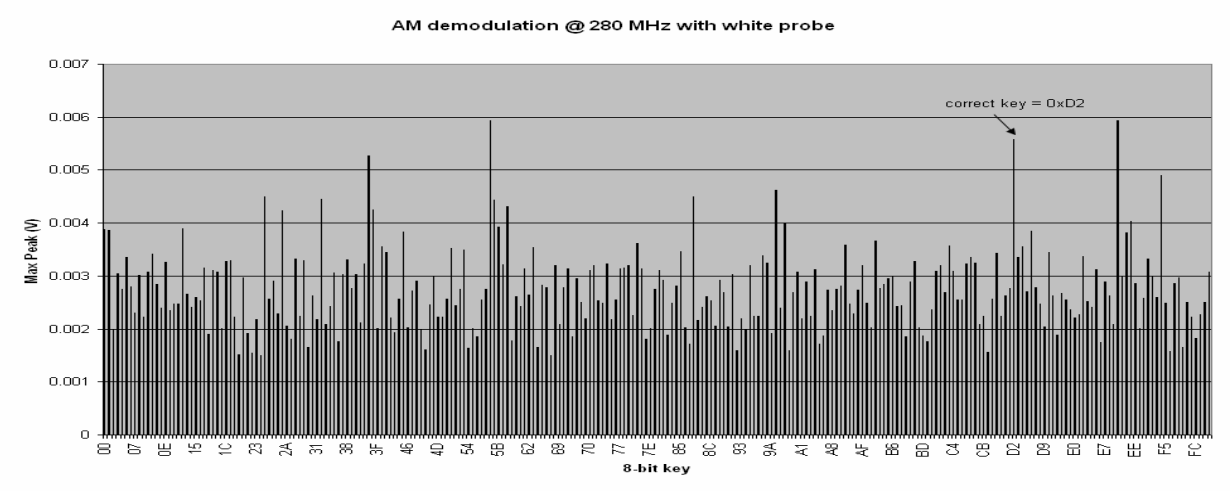
**a)**


DEMA with white probe
correct key = 0xD2
Max Peak (V)
8-bit key

**b)**

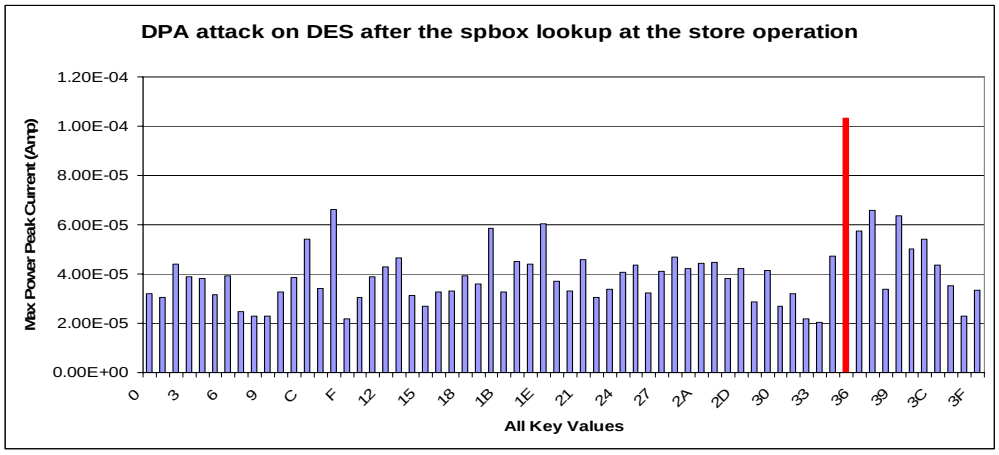

AM demodulation @ 360 MHz with commercial probe
correct key = 0xD2
Max Peak (V)
8-bit key

**c)**

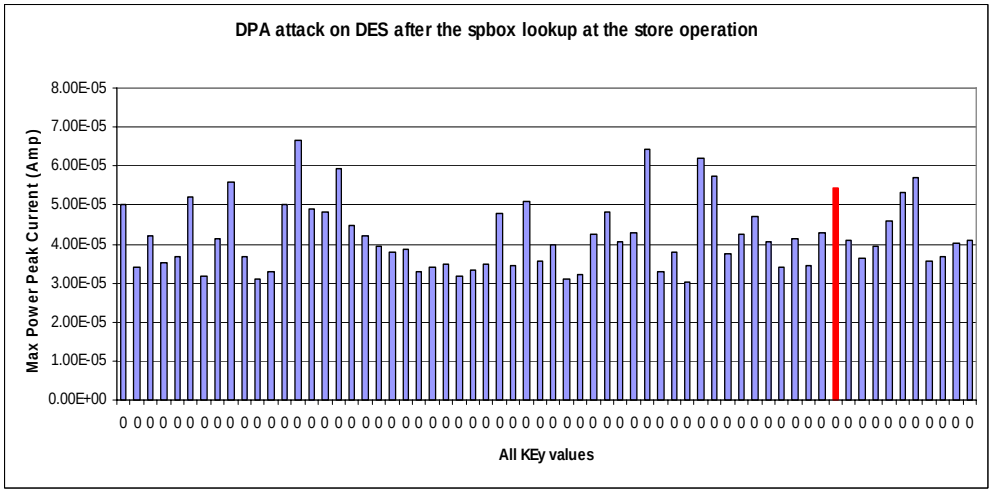



d)

**Fig. 11 Commercial vs hand-made probe results without and with demodulation.**



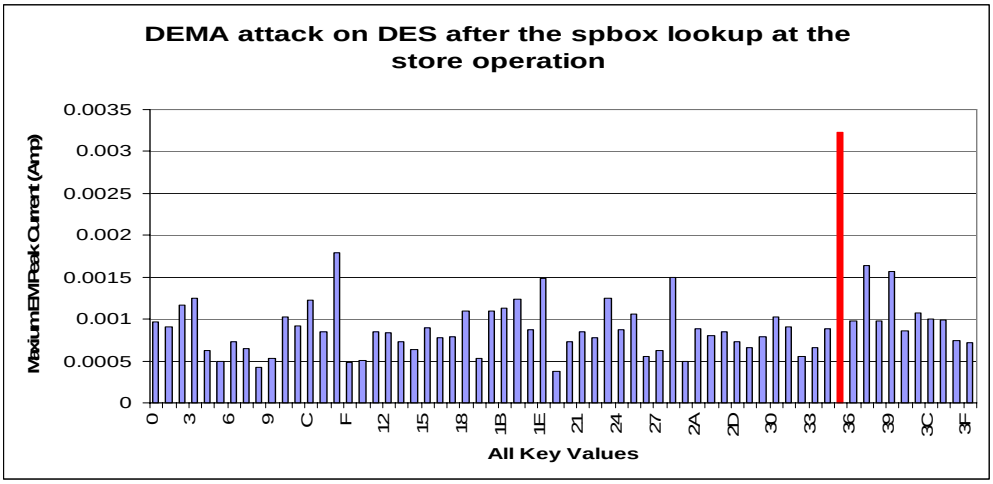

a)




b)

**Fig. 12. 1st order DPA on DES without and with Countermeasure in a) and b) respectively**

a)




b)

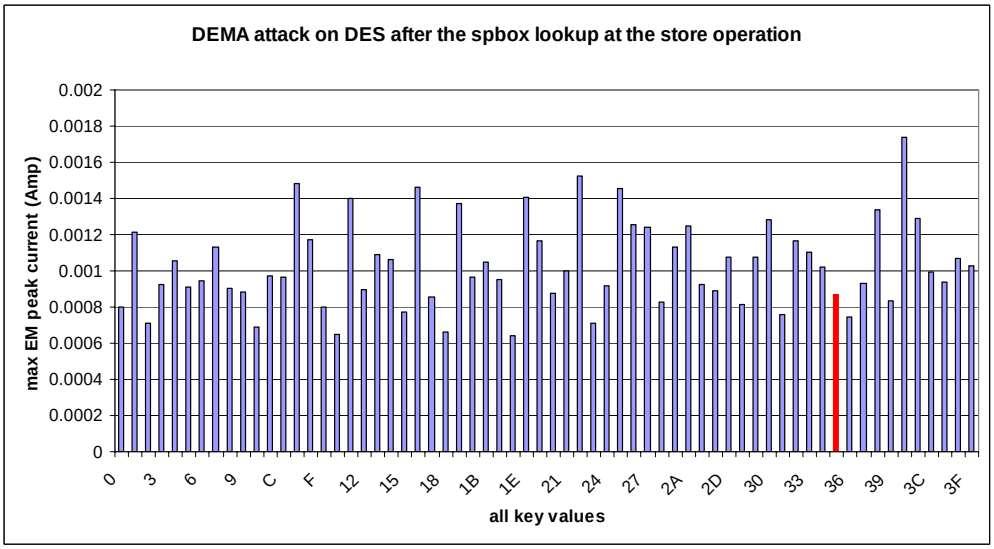



**Fig. 13. 1st order DEMA on DES without and with Countermeasure in a) and b) respectively**

A 2nd order differential EM analysis was also performed on Rijndael in order to see if the correct guessed key would be apparent. In the proposed countermeasure the Rijndael S-box was transformed into the masked S-box and the mask table which were accessed by the ARM processor for different plaintexts. In the analysis, three acquisitions of the same set of plaintext and same set of masked S-boxe tables and mask tables were performed. Each acquisition produced 6000 EM traces, 3000 were used to obtain samples of the masked S-box and the other 3000 were used to obtain the corresponding samples of the mask tables. Different bits at the output of the masked S-box load were used to perform the partitioning (see bit *j* in section 1.1). Unless otherwise stated, the probability difference, $\Lambda_{m=0}$ or $\Lambda_{m=1}$ (see Appendix A), was used to illustrate the results when the value of *m* was known to the attacker *(known m)*. The probabilities do not lie between 0 and 1, since they have not been normalized (see equations used in Appendices). Alternatively the absolute probability difference or statistic $\Lambda$ was used, which illustrates an attack where the value of *m* is unknown *(unknown m)*. The all keys guess analysis for known *m* is shown in figure 14a) using the least significant bit at the output of the S-box table for partitioning. In all cases the correct key is at 00 (the correct key has all bits equal to zero, so if the analysis is successful, then the highest peak in the bar chart would be at the far left at 00, identifying the correct key). Assuming the value of *m* is unknown the all keys guess analysis is shown in figure 14b) for

the same set of EM traces and same partitioning bit. The heuristic [15] for the 2nd order analysis presented in Appendix A was also used to plot the all keys guess and it is shown in figure 14d) for the same set of EM traces, using bit 3 to partition. Figure 14c) illustrates the successful results from the same set of EM traces, also using bit 3, however using the statistic. The statistic identified the correct key, whereas the heuristic found the correct key to be the second highest peak. Using a total of 1400 EM traces, 8 out of 32 partitioning bits revealed the correct key. When the total number of EM traces was increased to 2800 and 8000, the number of partitioning bits which revealed the correct key increased to 10 and 13 out of 32 respectively. However in all cases, a histogram of keys with the highest peaks in each analysis, identified the correct key. In other words of the keys with the highest peaks, the correct key was most frequently identified as correct from the analyses with all 32 partitioning bits. Analysis using different values of *m* and over 5600 EM traces using the statistic revealed the correct key in only 2 out of 32 analyses.

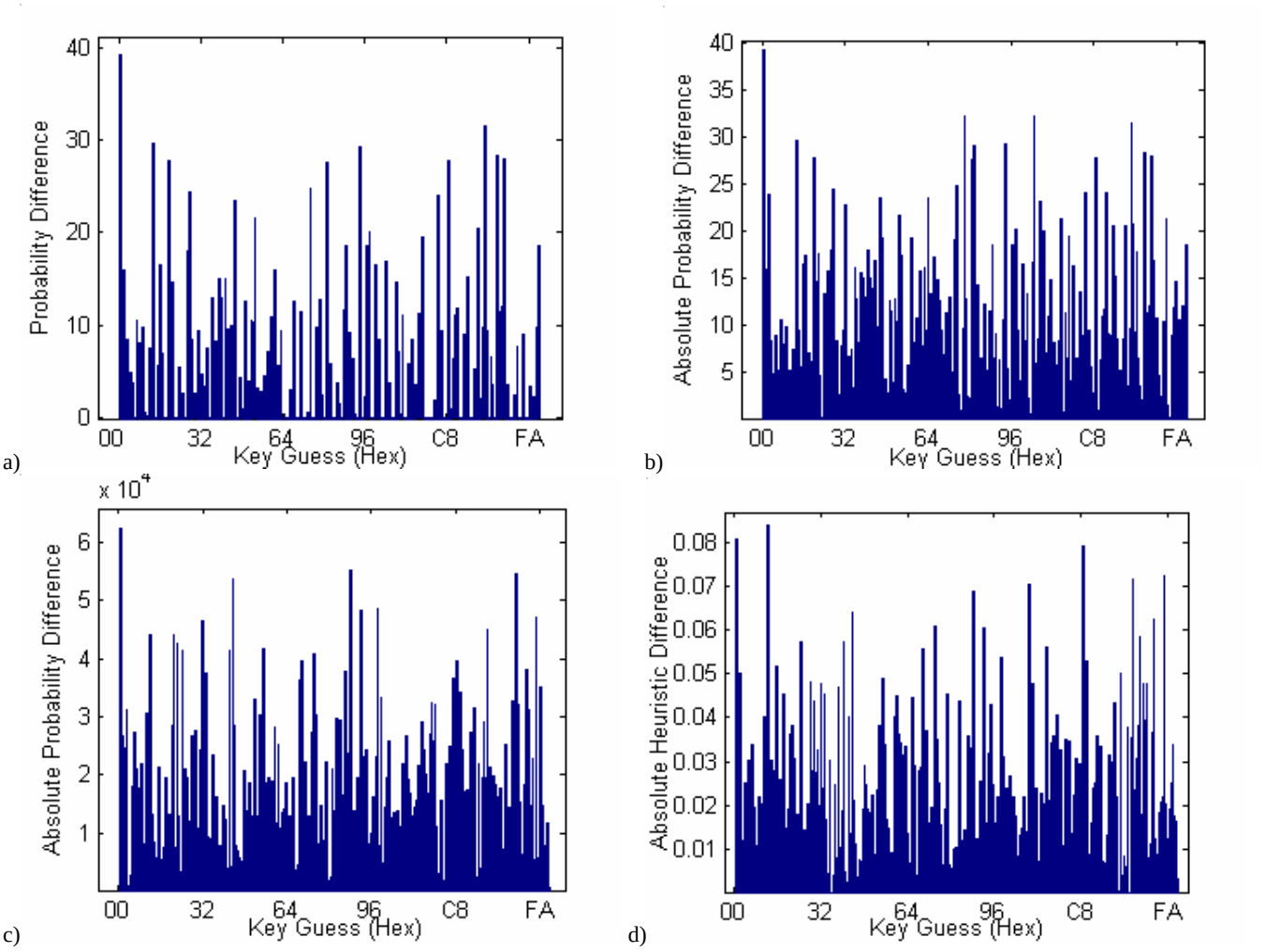
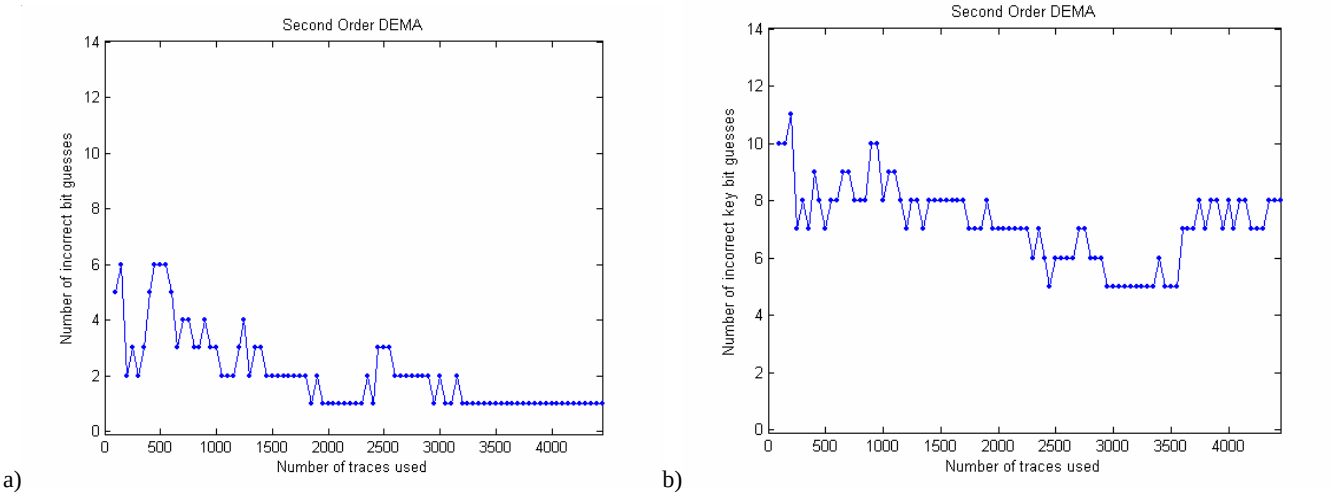



**Fig. 14. All keys guess using 2nd order DEMA statistic for known *m* and unknown *m* in a),b) respectively for bit 1, and unknown *m* with statistic and heuristic for bit 3 in c),d) respectively.**

The same two sets of EM traces was also used to analyze the convergence time for a 2nd order DEMA. In this experiment the analysis is performed to determine the value of *m* (analogous to the attack on the whitening or key-xoring countermeasure[15], except the analysis is done at the output of

the S-box and unlike [15] where the key is determined, here we determine the value of *m*). The y-axis represents the number of bits of *m* which were incorrectly predicted using the statistic in Appendix A. Again a different set of acquisitions was also run with different random masks, and the results are shown in figure 15b). In figure 15a) all 31 bits out of 32 bits of *m* (at the output of the S-box) were determined by the 2nd order DEMA. In figure 15b) only 24 bits out of 32 bits of *m* are correctly determined using this analysis. In figure 15 over 8000 EM traces were involved in the computations and figure 15a) uses a different set of masked S-box and mask tables than figure 15b).




a) b)

**Fig.15. Number of incorrect bits of *m* versus number of EM traces using second order DEMA using different random masks.**

A third order DEMA analysis (see appendix B) was also performed using the table masking countermeasure. In this experiment approximately 6000 EM traces using different random masks were acquired. Some results using unknown *m* are shown in Figure 16 a), b) and c) with same EM data but using bit 1,2, and bit 3 respectively as a partitioning bit. . In all cases the correct key is at 00 (the correct key has all bits equal to zero, so if the analysis is successful, then the highest peak in the bar chart would be at the far left at 00, identifying the correct key). Figure 16a) correctly identifies the correct key however figure 16 b) and c) do not. Figure 16c) is close, since the correct key is the fourth highest peak. Using over 5000 EM traces, only 1 out of 32 analyses, was able to find the correct key as the highest DEMA peaks. The histogram of these keys with the highest peaks resulting from the 32 analyses did not indicate the correct. In the second set of EM acquisitions, obtained from the same set of data, none of the analyses revealed the correct key.

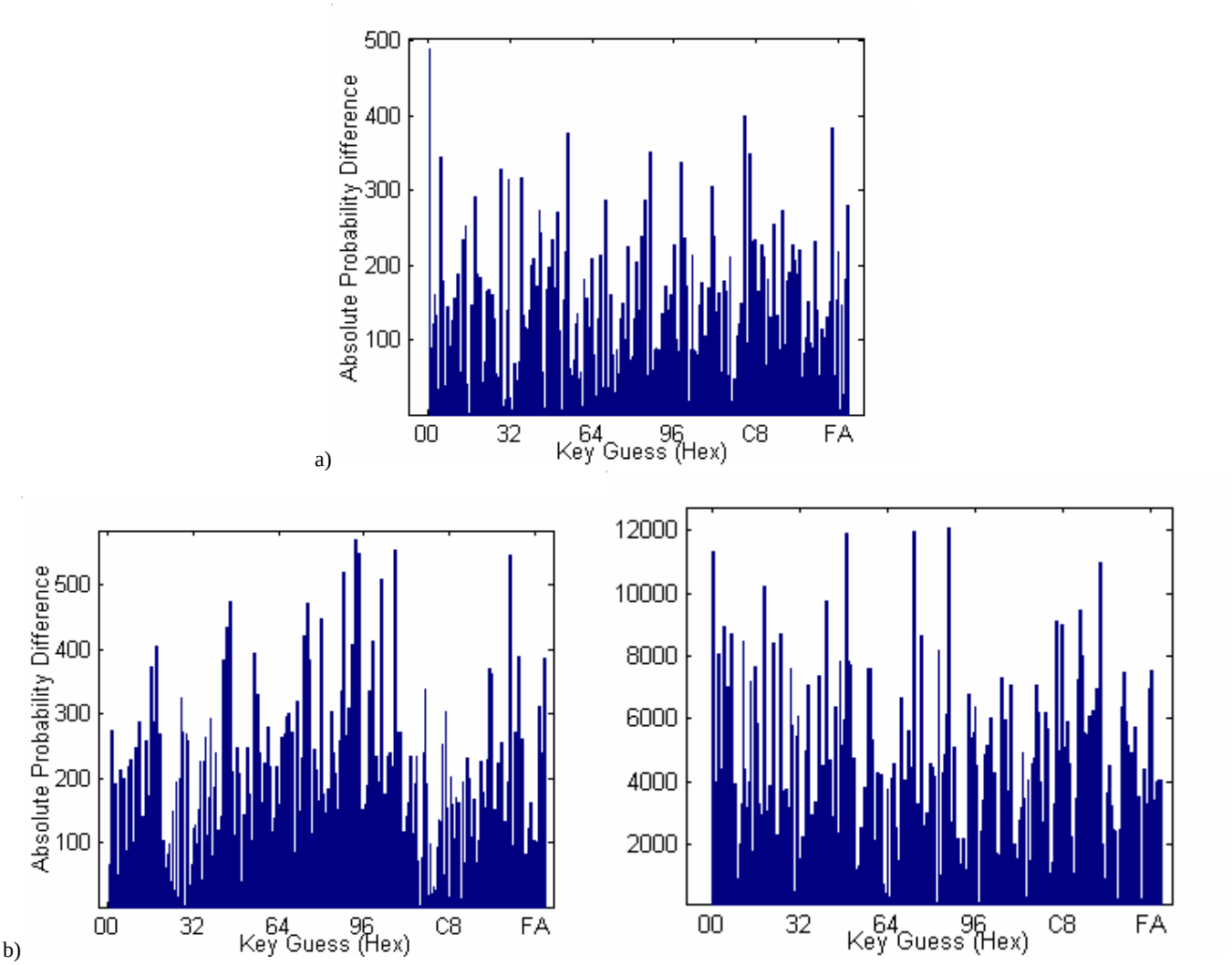
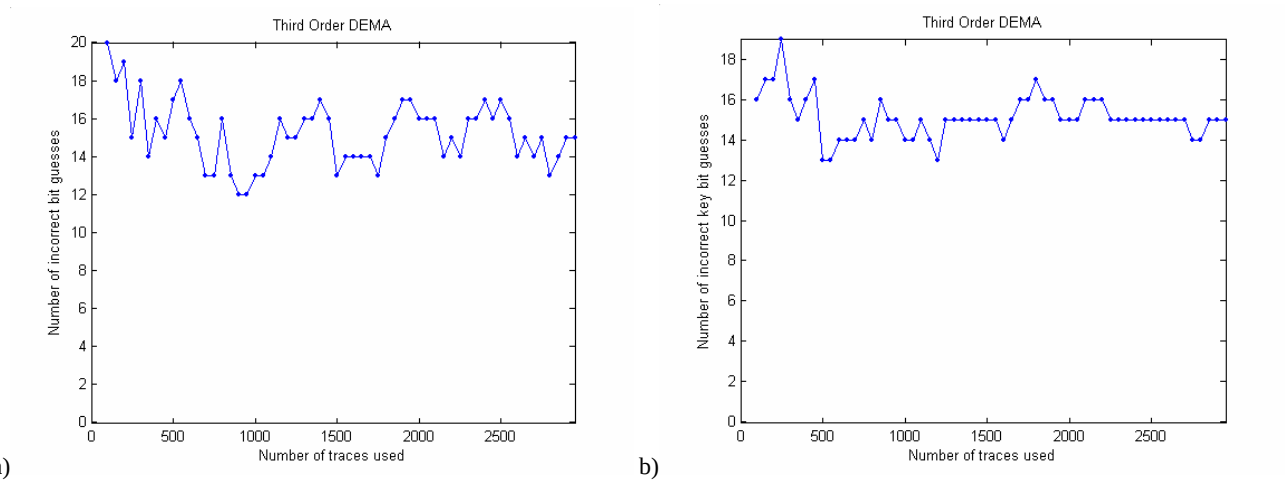



**Fig. 16. The 3rd order DEMA, showing all keys guess for unknown *m* using bit 1,2,3 in a),b),c) illustrating correct, incorrect and incorrect key guesses respectively.**

Figure 17 illustrates the convergence time for a third order differential EM attack on the value of *m*. In both cases, different random masks are used in addition to different acquisitions. However in both cases 15 out of 32 bits of the value of *m* remain incorrectly determined by the analysis. Both figures, use the same value of *m* but each involve 3 different sets of acquisitions and different masked S-box tables and mask table, each having a total of 6000 EM traces (since the x-axis reports the number of EM traces in each partition, for partition bit being a 0 and being a 1). It is interesting to note that the number of incorrect bits of *m* in figure 17 do not seem to decrease, but instead appear to converge at 15 incorrect bits.




a) b)

**Fig. 17. The 3rd order (split mask) DEMA for determining the value of *m* using different sets of randomly generated masks and data acquisitions in a) and b).**

The energies of previously researched countermeasures [5,14] were computed to compare with the proposed countermeasure energy. The evaluation board containing the 32-bit ARM7TDMI RISC processor core on one chip separate from the memory along with a current meter was used to obtain current measurements. All the energy measurements were obtained from a current meter measuring the current drawn from the supply pin of the chip core, while the processor was executing the algorithm in an inifinite loop. These energies reflected the processor core energy consumption only and not the input/output buffer power or memory power. Several different countermeasures were implemented in the ARM7TDMI in order to compare energies. The proposed countermeasure using only one additional table (thwarting a 1st order DEMA) is shown as *P1*, whereas the proposed countermeasure using 2 extra tables (thwarting a 2nd order DEMA, thus providing increased security) is shown as *P2*. The total number of 256 by 32-bit tables (denoted as *|S-box|)* used by the Rijndael algorithm, the number of memory accesses (*|ld/st|* for *load/store*) and the average current measured from the processor while it was executing the Rijndael ( *I(mA)* ) algorithm is illustrated in table 1. The original Rijndael algorithm, optimized for ARM7TDMI is shown in column 2 as *Rijndael*. Finally the energies (current multiplied by the supply voltage and latency of each implementation) is given in the table in *mJ* ($E_p$). Even the two table implementation of the proposed countermeasure (for example using $M_1[x]$ and $M_2[x]$, in place of *M[x]*), *P2*, is more energy efficient than [5], where table regeneration is required in order to change the random mask. The proposed countermeasure (*P1*) provided 1.7 times increase in energy (*E times*) over Rijndael with no countermeasure (*Rijndael)* and 5.2 times less energy than the countermeasure in [5] where table regeneration is required when a new mask is applied.

**Table 1. Comparison of Processor Energy**

| | *Rijndael* | *P1* | *P2* | [5] |
|---|---|---|---|---|
| *\|S-box\|* | 5 | 6 | 7 | 5 |
| *\| ld/st\|* | 160 | 320 | 480 | 2048 |
| *I (mA)* | 4.46 | 4.41 | 4.40 | 4.17 |
| $E_p$ *(mJ)* | 0.33 | 0.57 | 0.81 | 2.92 |
| *E times* | 1 | 1.7 | 2.4 | 8.9 |

**Table 2. Comparison of SRAM Dynamic Power**

| | *Rijndael* | *P1* | *P2* | [14] | [14] |
|---|---|---|---|---|---|
| *\|masks\|* | 0 | 256 | 256 | *2* | *50* |
| *HODEMA* | 1 | 2 | 3 | *~1* | *~1* |
| $P_m$*(mW)* | 2.26 | 4.92 | 7.97 | *3.24* | *51.4* |
| *P times* | 1 | 2.17 | 3.5 | *1.4* | *22.7* |

Since table 1 did not represent the energy dissipation of the memory, and since it is well known that memory energy dissipation is significant and often dominates, an analysis was performed with SRAM models from [31]. A comparison with the previously researched countermeasure which stores one set of S-box tables per mask[14], is given in table 2 which illustrates the impact of memory size on power dissipation. The number of different masks supported by each countermeasure is shown as *|masks|*, the order of the DEMA which may be required to attack the countermeasure (*HODEMA*) and the dynamic power dissipation of each SRAM ($P_m$ *(mW)* ) which would hold all the tables used in the Rijndael algorithm for each countermeasure are given. The proposed countermeasures (*P1, P2*) require significantly less memory when supporting the same number of masks. For example with $(1/5)^{th}$ the number of masks (last column), [14] dissipates up to 10.4 times more energy than the proposed countermeasure.

## 6. Discussion and Conclusions

This research presents for the first time a new table masking countermeasure and analysis using real EM measurements of a 32-bit embedded processor core, the ARM7TDMI. The energy for this countermeasure is compared to energies for implementation of other countermeasures. The proposed countermeasure provides random masking without the large energy overheads of table regeneration in [5] and requires significantly fewer tables than the countermeasure in [14], where one set of tables per mask is required. Similar to [5,18], the proposed countermeasure has the potential for attack through a $2^{nd}$ order DEMA, however unlike [5,18], the proposed countermeasure can increase the security by increasing the number of extra tables. The table masking countermeasure can trade off memory for security, thus thwarting a higher $n^{th}$ order DPA using *n* tables.

As evident from the experimental analysis with the proposed countermeasure in section 5, the $3^{rd}$ order DEMA was more difficult to launch than the $2^{nd}$ order DEMA. The statistic for the third order analysis also required more complex computations than the $2^{nd}$ order analysis. The all key guess experiments further indicated that more $2^{nd}$ order analyses revealed the correct key compared to the $3^{rd}$ order analyses. The set of random masks used in the tables, including the value of *m* may have an impact on the results in addition to the noise from the experimental acquisition of EM traces. However noise is clearly a significant factor as evident from $1^{st}$ order DEMAs, where significant DEMA peaks were evident in 100 EM traces in one acquisition and over 1000 EM traces in another acquisition of identical data. Results showed that demodulation of the EM signals did not improve the analysis, unlike experiments in [10]. This is likely because EM acquisition was done very close to the packaged chip (not in the far-field), hence the EM traces more closely resemble the power grid of the chip. Previous

research [5] used a constant mask, for all data in the table and when a new random masking is required, the complete set of S-box tables were regenerated[5] to utilize the next different random value mask. The proposed table masking countermeasure is similar to the duplication method [18] however unlike [18], the second table does not hold the random mask of the S-box entry, *r[x]*, and the duplication method [18] used two tables whereas this countermeasure can increase the number of tables thwarting higher order DEMA. It is interesting to note that according to our experimental results, when the value of *m* is changed periodically, the $2^{nd}$ order all keys analysis becomes almost as difficult as a $3^{rd}$ order all keys analysis.

This paper presented for the first time $2^{nd}$ and $3^{rd}$ order S-box DEMA results for a 32-bit low power embedded processor using real power and EM measurements with a new table masking countermeasure. This is unlike previous research which has investigated only $2^{nd}$ order DPA [15] or $2^{nd}$ order DEMA [10], both on 8-bit processors, or previous research which investigated theoretical variants of $2^{nd}$ order analysis [30] without real measurements. Results of the $3^{rd}$ order DPA/DEMA statistics on the new proposed table masking countermeasure found that a larger number of EM traces are required, making the table masking countermeasure very effective. The countermeasure requires negligible additional energy dissipation compared to previous countermeasures [5] and smaller storage overhead compared to [14]. The table masking countermeasure can trade off memory for security, thus thwarting a higher $n^{th}$ order DPA with *n* additional mask tables. This research is crucial for supporting low energy security for embedded systems which will be prevalent in wireless IP-enabled devices designed with nanometer technologies of the future. This research is supported in part by grants from NSERC, CITO and Industry.

**REFERENCES**

[1] P.Kocher “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems”, LNCS 1109, 1996.

[2] P.Kocher, J.Jaffe, B.Jun “Differential Power Analysis” Crypto’99, LNCS 1666, 1999

[3] J.Coron, P.Kocher, D.Naccache “Statistics and secret leakage” LNCS 1962, 2001, pp.157-173.

[4] T.Messerges, etal. “Investigations of Power analysis attacks on Smartcards” USENIX workshop on Smartcard Technology, 1999.

[5] T.Messerges, “Securing the Rijndael finalists against power analysis attacks” LNCS 1978, 2001, pp.150-164.

[6] H.Saputra, etal. “Masking the energy behavior of DES encryption”, Proceedings of DATE 2003.

[7] D.Agrawal et al. “The EM side-channel(s)” CHES 2002, 2002, pp.29-45.

[8] E.Biham, A.Shamir “Power analysis of the key scheduling of the Rijndael candidates”, $2^{nd}$ Rijndael conference, 1999.

[9] S.Chari, etal. “Towards sound approaches to counteract power-analysis attacks”, LNCS 1666, 1999, pp.398-412.

[10] D.Agrawal, etal. “The EM side-channel…methodologies” at http ://www.research.ibm.com/intsec/emf.html

[11] J.Daemen, V.Rijmen “Resistance against implementation attacks”, $2^{nd}$ Rijndael conference, 1999.

[12] M.Akkar, etal. “Power analysis, what is now possible…”, LNCS 1976, 2000, pp489-502.

[13] S.Chari etal. “A cautionary note regarding evaluation of Rijndael candidates on smart-cards”, $2^{nd}$ Encryptn. Std. Cand. Conf, March 1999.

[14] K.Itoh M.Takenaka, N.Torii “DPA countermeasure based on the masking method”, LNCS 2288, 2002,pp.440-456.

[15] T.Messerges “Using $2^{nd}$ Order Power Analysis to attack DPA resistant software”, LNCS 1965, 2000 pp.238-251.

[16] M.Aydos, T.Yanik, C.K.Koc “An high speed ECC-based wireless authentication protocol on an ARM Microprocessor”, Proc of $16^{th}$ Annual Comp. Sec. Appl. Conf. 2000.

[17] G.Hachez, F.Koeune, J.-J. Quisquater “cRijndaelar results: Implementation of four Rijndael candidates on two smart cards”, $2^{nd}$ Rijndael conference, 1999.

[18] L.Goubin, J.Patarin “DES and Differential power analysis- the duplication method” CHES 2001.

[19] J.Golic “Multiplicative Masking and power analysis of Rijndael”, CHES 2002.

[20] J.Daemen M.Peeters, G.V.Assche “Bitslice ciphers and power analysis attacks” Proc of Fast software encryption workshop, 2000.

[21] J.Coron, L.Goubin "On Boolean and arithmetic masking against differential power analysis", CHES 2000.

[22] J.Coron "Resistance against differential power analysis for ECC" CHES 1999.

[23] S.Mangaard, "A Simple Power-Analysis Attack on Implementations of the Rijndael Key Expansion", ICICS 2002 , LNCS 2587, pp.343-358, 2003.

[25] E.Trichina, L.Korkishko "Secure and efficient Rijndael software implementations for smart cards", Cryptology ePrint Archive, 2004/149 http://eprint.iacr.org/2004/149.pdf , 2004.

[26] E.Brier, C.Clavier, F.Olivier "Correlation power analysis with a leakage model" CHES 2004, LNCS 3156, pp16-29, 2004.

[27] J.Daemen, V.Rijmen "Rijndael Proposal: Rijndael" http://csrc.nist.gov/encryption/Rijndael, 1999.

[28] B. Gladman, "A Specification for Rijndael, the Rijndael Algorithm", at fp.gladman.plus.com /cryptography_technology /Rijndael /Rijndael.spec.311.pdf, 15 April 2003 (pages 18-19).

[29] K.Gandolfi etal. "Electromagnetic analysis: concrete results" , CHES 2001, LNCS 2162, pp.251-261, 2001.

[30] J.Waddle, D.Wagner "Towards efficient second-order power analysis" CHES 2004, LNCS 3156, pp1-15, 2004.

[31] W.Liao etal. "leakage power modeling and reduction with data retention", IEEE ICCAD, 2002, pp. 714-719

[32] K.Tiri etal. "A side channel leakage free coprocessor in .18u CMOS for embedded AES-based Cryptographic and biometric processing", DAC, 2005.

[33] C.Gebotys, A.Tiu, X.Chen "A Countermeasure for EM attack of a Wireless PDA" IEEE International Conference on Information Technology – Special Session on Embedded Cryptographic Systems, Las Vegas, AZ, April 2005, pp. 544-549

[34] C.Gebotys, S.Ho, A.Tiu "EM Analysis of Rijndael and ECC on a Wireless Java-based PDA" accepted for Publication in LNCS, CHES 2005 (15 pages).

[35] C.Gebotys "Low Energy Security Optimization in Embedded Cryptographic Systems", Proceedings of IEEE/ACM/IFIP ISSS-CODES, Stockholm, Sweden, September 2004, pp.224-229.

[36] C.Gebotys "Design of Secure Cryptography Against the Threat of Power-Attacks in DSP embedded processors", ACM Transactions on Embedded Computer Systems, Vol.3, No.1, February 2004, pp92-113.

## Appendix A: 2nd Order S-box Differential Analysis derivation

The following partial code will be used to derive the 2nd order S-box DPA. We assume that the power sample of the load of the corresponding mask table (*M[x]*) is available (or random mask used in full random masking case) as well as the power consumption of the masked S-box table load (*S'[x]*, where *x* is an 8-bit quantity, in Rijndael $x=mask \oplus plaintext \oplus secret\_key = m \oplus p \oplus k$ ).

...

8. load $S'[(m \oplus p \oplus k)_{0..7}]$ ; where $S'[m \oplus p \oplus k] = S[k \oplus p] \oplus r_i$

*a.* load $M[(m \oplus p \oplus k)_{0..7}]$

Let the k[th] normalized power consumption of the random data (line a.) ($M[(m \oplus p \oplus k)_{0..7}]$) be represented by $b1_k$, and the normalized power consumption of the masked output of the S-box table ($S'[(m \oplus p \oplus k)_{0..7}]$, on line 8) be $c_k$ . Let x=$(m \oplus p \oplus k)_{0..7}$ . Let $M[(m \oplus p \oplus k)_{0..7}]_i$ and $S'[(m \oplus p \oplus k)_{0..7}]_i$ be the *ith* bit of $M[(m \oplus p \oplus k)_{0..7}]$ and $S'[(m \oplus p \oplus k)_{0..7}]$ respectively. The distribution for these values is assumed to be Gaussian with mean of zero and standard deviation of one (due to normalization, $N(\mu,\sigma) = N(0,1)$) $f_{b1}(b1_k) \approx N(0,1)$ , and $f_c(c_k) \approx N(0,1)$ . Let $f_{b1}^{-} = f_{b1}(b1_k \mid \mathrm{M}[x]_i = 0) \approx N(-\frac{\varepsilon}{2},\sigma)$ represent the distribution of the power consumption values of $b1_k$ such that the i[th] bit of *M[x]* is zero. Assume it is also a Gaussian distribution. Similarly let the following

distribution of power consumption exist $f_c^- = f_c(c_k \mid S'[x]_i = 0) \approx N(-\frac{\varepsilon}{2}, \sigma)$ . Let $f_{b1}^+ = f_{b1}(b1_k \mid M[x]_i = 1) \approx N(+\frac{\varepsilon}{2}, \sigma)$ represent the distribution of the power consumption values of $b1_k$ such that the i[th] bit of *M[x]* is one. Similarly let the following distributions of power consumption exist $f_c^+ = f_c(c_k \mid S'[x]_i = 1) \approx N(+\frac{\varepsilon}{2}, \sigma)$ . Next one can calculate the following joint conditional probability distributions of $b1_k$ and $c_k$ (and $b1_k$ is equally likely to be a 0 or 1, and where the final mask , m, is equally likely to be a zero or a one):

$$f_{b1,b2,c}(b1_k, c_k \mid m \oplus S[x]_i = 1) = \frac{1}{2}(f_{b1}^- f_c^+ + f_{b1}^+ f_c^-)$$

$$f_{b1,b2,c}(b1_k, c_k \mid m \oplus S[x]_i = 0) = \frac{1}{2}(f_{b1}^- f_c^- + f_{b1}^+ f_c^+)$$

next one can substitute using the normal Gaussian distribution [15] to obtain:

$$\Pr(\Psi \mid m \oplus S[x]_i = 1) \approx \prod_{k=0}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k - c_k))\right]$$

$$\Pr(\Psi \mid m \oplus S[x]_i = 0) \approx \prod_{k=0}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k + c_k))\right]$$

So far our derivation is identical to that in [15] however here it is applied to our masked S-box scheme DPA. Now we will extend these result further to derive a statistic we can use in our analysis. The statistic will be used to determine the estimated probability of a correct key guess. We assume the user has control over the input plaintexts, hence based upon the attackers guess of 8 key bits, the attacker can determine the output of the S-box table. Using the following notation, let $\delta 1(key) = \{k \mid S[key \oplus p]_l = 1, \text{or } S'[key \oplus p]_l \oplus M[key \oplus p]_l = 1, \text{where } m_l = 0\}$ and $\delta 0(key) = \{k \mid S[key \oplus p]_i = 0, \text{or } S'[key \oplus p]_i \oplus M[key \oplus p]_i = 0 \text{ where } m_i = 0\}$ to support partitioning of power traces (based upon the users input plaintext and 8 key bits guess), we obtain the following statistics:

$$\Delta_{m_i=0} \approx \prod_{k=0|k\in\delta 1}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k - c_k))\right] \times \prod_{k=0|k\in\delta 0}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k + c_k))\right]$$

$$\Delta_{m_i=1} \approx \prod_{k=0|k\in\delta 0}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k - c_k))\right] \times \prod_{k=0|k\in\delta 1}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k + c_k))\right]$$

Which represents the estimated probability that the partitioning of power traces according to the key guess is correct, given the value of the $j^{th}$ bit of *m*, $m_j$. Alternatively one could also use the following statistic representing the difference of estimated probabilities (probability of the partition being correct minus the probability of the partition being incorrect) given as the two formulas below depending upon the bit value of *m*:

$$\Lambda_{m_i=0} \approx \prod_{k=0|k\in\delta 1}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k - c_k))\right] \times \prod_{k=0|k\in\delta 0}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k + c_k))\right] - \prod_{k=0|k\in\delta 0}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k - c_k))\right] \times \prod_{k=0|k\in\delta 1}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k + c_k))\right]$$ **, for known *m***

$$\Lambda_{m_i=1} \approx \prod_{k=0|k\in\delta 0}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k - c_k))\right] \times \prod_{k=0|k\in\delta 1}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k + c_k))\right] - \prod_{k=0|k\in\delta 1}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k - c_k))\right] \times \prod_{k=0|k\in\delta 0}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k + c_k))\right]$$ **, for known *m***

However the difficulty lies in the fact that the attacker does not know the value of *m.* Hence a new statistic is required, as follows:

$$\Lambda \approx \left| \prod_{k=0|k\in\delta 0}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k - c_k))\right] \times \prod_{k=0|k\in\delta 1}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k + c_k))\right] - \prod_{k=0|k\in\delta 1}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k - c_k))\right] \times \prod_{k=0|k\in\delta 0}^{N-1}\left[2\cosh(\frac{\varepsilon}{2\sigma^2}(b1_k + c_k))\right] \right|$$ **, for unknown *m***

This last statistic, $\Lambda$, is independent of the value of *m.* However it assumes that *m* is some fixed value for all the power or EM traces acquired and that the probability of the partitioning corresponds to $m_j$ is higher than the probability of the partitioning corresponds to $\overline{m_j}$. If the attacker does not know the value of *m* the attack is more difficult since all probabilities are positive according to $\Lambda$, unlike the case where an attacker could rule out cases where $\Lambda_{m_j=0}$ is negative if the attacker knows that $m_j=0$. One could also use the heuristic from [15], as follows:

$$heuristic = \left| \frac{\sum_{k=0|k\in\delta 0} |b1_k - c_k|}{N} - \frac{\sum_{k=0|k\in\delta 1} |b1_k - c_k|}{N} \right|$$

## Appendix B: Derivation of 3rd order S-box Differential Analysis Statistic

The third order statistic can be derived using the same approach as illustrated in Appendix A. It will be used for the case where three tables are in use, specifically *S'[x]*, $M_1[x]$, and $M_2[x]$ (where $m = r[x] \oplus M_1[x] \oplus M_2[x]$, *or* $m \oplus S[x] = S'[x] \oplus M_1[x] \oplus M_2[x]$).

Line 1. load *S'[x]*

Line 2. load $M_1[x]$

Line 3. load $M_2[x]$

Let the k[th] normalized power consumption of the masked S-box *(line 1.)*, the first mask table *(line 2.)*, and the second masked table *(line 3.)*, be represented by $c_k$, $b1_k$, and $b2_k$ respectively. The distribution for these values is assumed to be Gaussian with mean of zero and standard deviation of one (due to normalization, $N(\mu,\sigma) = N(0,1)$) $f_{b1}(b1_k) \approx N(0,1)$, $f_{b2}(b2_k) \approx N(0,1)$, and $f_c(c_k) \approx N(0,1)$. Continuing with the same assumptions in Appendix A we obtain the following statistic:

$$f_{b1,b2,c}(b1_k, b2_k, c_k \mid S[x]_i \oplus m_i = 1) = \frac{1}{4}(f_{b1}^- f_{b2}^- f_c^+ + f_{b1}^+ f_{b2}^+ f_c^+) + \frac{1}{4}(f_{b1}^- f_{b2}^+ f_c^- + f_{b1}^+ f_{b2}^- f_c^-)$$

$$f_{b1,b2,c}(b1_k, b2_k, c_k \mid S[x]_i \oplus m_i = 0) = \frac{1}{4}(f_{b1}^- f_{b2}^- f_c^- + f_{b1}^+ f_{b2}^+ f_c^-) + \frac{1}{4}(f_{b1}^- f_{b2}^+ f_c^+ + f_{b1}^+ f_{b2}^- f_c^+)$$

next one can substitute using the normal distribution : $\eta(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$ and factor out the constant $\frac{1}{\sigma\sqrt{2\pi}}$ to obtain the following expression for the probability :

$$\Pr(\Psi \mid S[x]_i \oplus m_i = 1)$$

$$\approx \prod_{k=0}^{N-1} \left[ \begin{array}{l} \left( e^{-\frac{1}{2\sigma^2}\left((b1_k+\frac{\varepsilon}{2})^2+(b2_k+\frac{\varepsilon}{2})^2+(c_k-\frac{\varepsilon}{2})^2\right)} \right) + \left( e^{-\frac{1}{2\sigma^2}\left((b1_k-\frac{\varepsilon}{2})^2+(b2_k-\frac{\varepsilon}{2})^2+(c_k-\frac{\varepsilon}{2})^2\right)} \right) \\ + \left( e^{-\frac{1}{2\sigma^2}\left((b1_k+\frac{\varepsilon}{2})^2+(b2_k-\frac{\varepsilon}{2})^2+(c_k+\frac{\varepsilon}{2})^2\right)} \right) + \left( e^{-\frac{1}{2\sigma^2}\left((b1_k-\frac{\varepsilon}{2})^2+(b2_k+\frac{\varepsilon}{2})^2+(c_k+\frac{\varepsilon}{2})^2\right)} \right) \end{array} \right]$$

For the rest of the analysis let $\varepsilon$ represent $\frac{\varepsilon}{2\sigma^2}$

$$\Pr(\Psi \mid S[x]_i \oplus m_i = 1)$$

$$\approx \prod_{k=0}^{N-1} \left( e^{-(\varepsilon b1_k + \varepsilon b2_k - \varepsilon c_k)} \right) + \left( e^{-(-\varepsilon b1_k - \varepsilon b2_k - \varepsilon c_k)} \right) + \left( e^{-(\varepsilon b1_k - \varepsilon b2_k + \varepsilon c_k)} \right) + \left( e^{-(-\varepsilon b1_k + \varepsilon b2_k + \varepsilon c_k)} \right)$$

$$\approx \prod_{k=0}^{N-1} \left[ \left( e^{-\varepsilon c_k} \left( e^{-(\varepsilon b1_k - \varepsilon b2_k)} + e^{-(-\varepsilon b1_k + \varepsilon b2_k)} \right) \right) + \left( e^{+\varepsilon c_k} \left( e^{-(\varepsilon b1_k + \varepsilon b2_k)} + e^{-(-\varepsilon b1_k - \varepsilon b2_k)} \right) \right) \right]$$

Next we substitute using the following trigonometric identity $\cosh(x) = \left( \frac{e^x + e^{-x}}{2} \right)$ to obtain the following:

$$\Pr(\Psi \mid S[x]_i \oplus m_i = 1)$$

$$\approx \prod_{k=0}^{N-1} \left( e^{-\varepsilon c_k} \left( e^{(-\varepsilon b1_k + \varepsilon b2_k)} + e^{-(-\varepsilon b1_k + \varepsilon b2_k)} \right) \right) + \left( e^{+\varepsilon c_k} \left( e^{-(\varepsilon b1_k + \varepsilon b2_k)} + e^{(\varepsilon b1_k + \varepsilon b2_k)} \right) \right)$$

$$\approx \prod_{k=0}^{N-1} (e^{-\varepsilon c_k} \cosh(\varepsilon(-b1_k + b2_k)) + e^{\varepsilon c_k} \cosh(\varepsilon(b1_k + b2_k)))$$

$$\Pr(\Psi \mid S[x]_i \oplus m_i = 0)$$

$$\approx \prod_{k=0}^{N-1} (e^{-\varepsilon c_k} \cosh(\varepsilon(b_k 1 + b2_k)) + e^{+\varepsilon c_k} \cosh(\varepsilon(-b1_k + b2_k)))$$

Now one can further use the following trigonometric identities $\cosh(x+y) = \cosh(x)\cosh(y) + \sinh(x)\sinh(y)$, $\sinh(-x) = -\sinh(x), \cosh(-x) = \cosh(x)$, and $\sinh(x) = \left(\frac{e^x - e^{-x}}{2}\right)$, so the equation can be rewritten as:

$$\Pr(\Psi \mid S[x]_i \oplus m_i = 1)$$

$$\approx \prod_{k=0}^{N-1}\left[e^{-\varepsilon c_k}(\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k) - \sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)) + e^{\varepsilon c_k}(\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k) + \sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k))\right]$$

$$\approx \prod_{k=0}^{N-1}\left[(e^{-\varepsilon c_k} + e^{\varepsilon c_k})\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k) + (-e^{-\varepsilon c_k} + e^{\varepsilon c_k})\sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)\right]$$

$$\approx \prod_{k=0}^{N-1}\cosh(\varepsilon c_k)\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k) + \sinh(\varepsilon c_k)\sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)$$

$$\Pr(\Psi \mid S[x]_i \oplus m_i = 0)$$

$$\approx \coprod_{k=0}^{N-1}\left[\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k)\cosh(\varepsilon c_k) - \sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)\sinh(\varepsilon c_k)\right]$$

Next we can further extend the expressions to directly solve for probabilities. Using the following notation, let $\delta 1(key) = \{k \mid S[key \oplus p]_i = 1, \text{or } S'[key \oplus p]_i \oplus M_1[key \oplus p]_i \oplus M_2[key \oplus p]_i = 1, \text{where } m_i = 0\}$ and $\delta 0(key) = \{k \mid S[key \oplus p]_i = 0, \text{or } S'[key \oplus p]_i \oplus M_1[key \oplus p]_i \oplus M_2[key \oplus p]_i = 0 \text{ where } m_i = 0\}$ to support partitioning of power traces (based upon the users input plaintext and 8 key bits guess), we obtain the following statistics:

$$\Delta_{m_i=1} \approx \prod_{k|k\in\delta 1}(\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k)\cosh(\varepsilon c_k) - \sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)\sinh(\varepsilon c_k))) \times$$

$$\prod_{k|k\in\delta 0}(\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k)\cosh(\varepsilon c_k) + \sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)\sinh(\varepsilon c_k)))$$

$$\Delta_{m_i=0} \approx \prod_{k|k\in\delta 0}(\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k)\cosh(\varepsilon c_k) - \sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)\sinh(\varepsilon c_k))) \times$$

$$\prod_{k|k\in\delta 1}(\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k)\cosh(\varepsilon c_k) + \sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)\sinh(\varepsilon c_k)))$$

**, for known *m***

Alternatively one can represent the statistic $\Lambda$ as given below, where the statistic is positive or negative to indicate the $m_i$ value is a 1 or 0 respectively.

$$\Lambda = \left|\Delta_{m_i=1} - \Delta_{m_i=0}\right|$$

$$= \left| \begin{array}{l} \prod_{k|k\in\delta 1} (\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k)\cosh(\varepsilon c_k) - \sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)\sinh(\varepsilon c_k))) \times \\ \prod_{k|k\in\delta 0} (\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k)\cosh(\varepsilon c_k) + \sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)\sinh(\varepsilon c_k))) \\ - \prod_{k|k\in\delta 0} (\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k)\cosh(\varepsilon c_k) - \sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)\sinh(\varepsilon c_k))) \times \\ \prod_{k|k\in\delta 1} (\cosh(\varepsilon b1_k)\cosh(\varepsilon b2_k)\cosh(\varepsilon c_k) + \sinh(\varepsilon b1_k)\sinh(\varepsilon b2_k)\sinh(\varepsilon c_k))) \end{array} \right| \textbf{ for unknown } \boldsymbol{m}$$

Note that if the value of *m* changes for each trace, it may not be possible to formulate a statistic unless we have a sample of *m* in which case it will lead to an even higher order statistic.