Low Energy Security Optimization in Embedded Cryptographic Systems

Catherine H. Gebotys
Dept of Elec and Comp Engineering
University of Waterloo
Waterloo, Ont, Canada
(519)885-1211
cgebotys@uwaterloo.ca

ABSTRACT

Future embedded and wireless devices will be increasingly powerful supporting many applications including one of the most crucial, security. Although many wireless and embedded devices offer more resistance to bus probing attacks due to their compact size, susceptibility to power/electromagnetic attacks must be analyzed. This paper presents optimized synthesis of new low energy masking countermeasures into cryptographic software. In particular a model for key masking with the objective of minimizing energy overhead is presented. Experimental results using real power measurements are shown to support up to 2.5 energy overhead savings and improved security compared to previous research. With the emergence of security applications in PDAs, cell phones, line card accelerators, etc, optimizing low energy countermeasures for resistance to power/ electromagnetic attacks is crucial for supporting future secure embedded devices.

Categories and Subject Descriptors

C.3 [**Special Purpose and Application Based Systems**].

General Terms

Design, Security.

Keywords

Power Analysis, Smart Cards, Embedded, Countermeasure.

1. INTRODUCTION

A wide range of embedded security will proliferate in automobile electronics, security for IP core protection in FPGAs and ASIC technologies, wireless devices such as PDAs, cell phones, and other areas. The cryptographic algorithms which are essential for these applications are typically run by embedded processors. As more security applications migrate to the wireless device, low energy resistance to attacks on the PDA or cell phone will become a necessity. Other embedded systems, such as line card accelerators, VPN systems, require high performance security. 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]. These attack resistant algorithms have been developed for smartcard applications, where energy dissipation or high performance is not viewed as important. 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. Researchers have already demonstrated that this new attack is viable[8]. 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. Both low energy and high performance constraints are often not considered in smart card research, where the main objective is to provide highest resistance to power analysis attacks or tampering at the lowest cost. Outside of smartcard research (which typically has been in the past limited to cheaper 8 bit or 16 bit processors), 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 is the ARM which is suitable for portable devices ranging from game devices to PDAs to cryptographic applications [5]. There is an important need to study energy optimized countermeasures for wireless portable devices, constrained by energy, or other embedded devices constrained by performance.

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]. This research studied the correlation of power variation with data values being manipulated and instruction sequencing[6]. For example hamming weight attacks have been studied by correlating the power for loading data with the hamming weight of the data[4]. In the former case, known as differential power attacks (DPA). Differential power attacks of embedded low power processors have not been reported in the literature. Higher order differential attacks[10] are an extension of the 1st order DPA which involve using joint statistics on multiple points within power traces.

Some countermeasures to these attacks have been suggested such as secret splitting[7], and random masking[5]. Secret splitting, involves splitting the secret data into smaller pieces and combining them with random data [7]. Then the cryptographic

CODES+ISSS'04, September 8–10, 2004, Stockholm, Sweden.

algorithm is run on each word, which is composed of random and secret data. When the table look ups (know as Sbox tables) are encountered, the key splitting approach requires the tables to be larger and requires an extra table. In the masking countermeasure, each secret piece of data is exclusive-or'd with a random data value (called a mask). Remasking the tables (or exclusive-or each table entry with a mask) within the algorithm is performed when the mask changes. In [5] remasking is suggested for every invocation of the algorithm (to thwart hamming weight attacks), whereas in [9] it is suggested for each round key. Overheads in latency[5] have been reported up to 1.9 times for AES (a standardized encryption algorithm). These overheads are largely due to remasking of tables, so some researchers have investigated storing a limited number of masked tables [9]. The authors define the terms key XORing DPA and Sbox DPA. The key XORing DPA was an attack on the result of exclusive oring the plaintext with the (masked or unmasked) key. The Sbox DPA was an attack on the (masked or unmasked) output of the Sbox table. In both attacks it is assumed that the attacker has control over the input plaintexts which are exclusive or'd with the key to index the Sbox table. However results using real power measurements were not performed. Later a second order DPA attack was developed [10] using the key XORing or "data whitening" as an example. A second order is required since the power sample of the mask and the power sample of the XOR result are used (in an n^{th} order DPA n power samples are required). Other research has avoided the use of table to support masking during key generation[14], however the use of tables provides significant performance improvements. Even though cryptographic algorithms often have symmetry, masking through key generation is difficult to perform by hand, due to the way round keys are generated and manipulated. Researchers have explored splitting the secret data, however splitting the masking data which is needed to obtain the secret data has not been explored.

Although integer linear programming (ILP) optimization is NP complete in general, it becomes practical in many instances when the problems have structure (such as network flow, matching, and other problems). Automatic mapping into a ILP can be supported with logical inferences. In particular if the ILP can be formulated from all Horn clauses, then this model can be solved optimally in polynomial time[11]. However it has also been shown that many logical inference models can be solved as a relaxed LP even though they are not Horn clauses[12].

Power analysis countermeasures must be energy optimized and/or performance optimized for many embedded systems such as wireless device implementation or VPN accelerators. A methodology for incorporating optimized masking countermeasures for cryptographic algorithms is presented with the objective of minimizing the energy overhead. The key generation phase is extracted from the algorithm and an ILP model is generated using logical inferences. The energy overhead is minimized and the model solves for the best masking with high order security. Attacks are discussed using high order DPAs as well as hamming weight information which an attacker may possibly obtain. Constraints to support mask randomization can also be incorporated as well as loop and conditional branch support. Unlike previous research, masks are split (split masks) through key generation such that the final mask on the round keys are never explicitly computed, enforcing a higher order security. Significant energy overhead is saved by eliminating the need to remask tables within the cryptographic algorithm.

2. METHODOLOGY FOR SECURITY

This section will describe the methodology for synthesizing the low energy masking countermeasure into a cryptographic application. Initially this methodology focuses on the key generation stage of the cryptographic application (however in theory it could be extended into the encryption stage as well). The methodology first determines the lower bound on the memory overhead and energy overhead for implementing the masking countermeasure on the cryptographic application. It then investigates constraints on the memory size and minimizes the total energy overhead of the countermeasure. If the energy overhead is too high, the memory size is increased and the optimization problem is rerun. The methodology utilizes an optimization model to obtain bounds and synthesize the masking solutions. The cryptographic application or key generation task is first converted into propositional logic. Next the logic is converted into the conjunctive normal form and finally into the linear mathematical form. The logical variables are replaced by binary variables. This will be further illustrated in section 3.

To provide key security, the master key is initially masked with a set of masks (since the key is typically 128bits or more). Masks are then introduced throughout the key generation (which generates numerous round keys to be used later in encryption/decryption). A mask set is generated from an initial set of masks (masks of the master key) by combining all possible masks to create new masks. The final round keys are constrained to have a fixed mask. In the model the user can fix this value as a combination of given masks within the mask set or the model can determine the best fixed final mask. If the user fixed this value, security constraints can be added to ensure resistance to 1st and 2nd DPA attacks. This final fixed mask for the round keys supports the use of one masked table to be used during encryption (without requiring energy expensive table regeneration). The optimization problem minimizes the additional operations (masking, loading of masks, conversion of masks,etc). In general both boolean masking and arithmetic masking may be required depending upon the operations within the key generation algorithm. Both are supported in this methodology and conversion between the maskings are also accounted for in the energy objective.

This paper will illustrate the methodology using Boolean masking within Rijndael[15] (or AES) a current standard for encryption. In Rijndael the 128 bit master key, represented by *rk(0), rk(1), rk(2), rk(3)* is input to the key generation algorithm. The key generation algorithm derives all round keys *rk(4), rk(5),…,rk(43)* for use in the Rijndael encryption algorithm. The following code represents part of the key generation followed by part of the encryption algorithm (using Tables *Te0-3*), where $\oplus$ represents the exclusive or operation:

For i = 0 to4{

rk(8i+4) = keyrotate(rk(8i+3) ⊕ rcon(i) ⊕ rk(8i+0)

rk(8i+5) = rk(8i+1) ⊕ rk(8i+4)

rk(8i+6) = rk(8i+2) ⊕ rk(8i+5)

rk(8i+7) = rk(8i+3) ⊕ rk(8i+6) … }

… s0 = pt ⊕ rk0 ; s1 = pt1 ⊕ rk1 ; …

t0=Te0(s0>>24) ⊕ Te1(s1>>16) ⊕ Te2(s2>>8) ⊕ Te3(s3) ⊕ rk4

…

s0 =Te0(t0>>24) ⊕ Te1(t1>>16) ⊕ Te2(t2>>8) ⊕ Te3(t3) ⊕ rk8

…

Since all round keys will have a fixed value of mask, table regeneration is not required (since tables are precomputed once for all keys). By eliminating the table regeneration, the Rijndael encryption will have high performance and significantly lower energy dissipation. This section will describe a model for automating the masking process such that energy overhead is minimized. The model will be illustrated for AES key generation which is one of the most complex key generation algorithms, as described in the previous section. The model will determine the optimal values of masks (shown below as *mmasked(), mfinal()*) such that the energy overhead is minimized. For example the AES example now becomes generalized (with masked tables *masked_Te0-3* being accessed by all round keys)

For i = 0 to 4{
rk(8i+4) = keyrotate(rk(8i+3) ⊕ rcon(8i+0) ⊕ rk(8i+0)
rk(8i+0) ⊕ =mfinal(0) ; rk(8i+4) ⊕ =mmasked(4)
rk(8i+5) = rk(8i+1) ⊕ rk(8i+4)
rk(8i+1) ⊕ =mfinal(1) ; rk(8i+5) ⊕ =mmasked(5)
rk(8i+6) = rk(8i+2) ⊕ rk(8i+5)
…rk(8i+7)=rk(8i+3) ⊕ rk(8i+6) …}
… s0 = pt ⊕ rk(0) ; s1 = pt1 ⊕ rk(1) ; …
t0 = masked_Te0(s0>>24) ⊕ masked_Te1(s1>>16) ⊕ masked_Te2(s2>>8) ⊕ masked_Te3(s3) ⊕ rk4; …

The next section will introduce the notation, model and extensions for supporting optimized masking for energy, memory and security constraints.

3. MODEL FOR ENERGY MASKING

Let $rk = \{0,1,2,...n\}$ represents the set of n round keys, $type = \{original, masked, final\}$ represents the type of round key (original, masked or final masked), and $mask = \{m0, m1, ..., mi, ..., mm, mnull\}$ represents the set of $m+1$ masks to be considered (where *mnull* represents no mask, for example in boolean and arithmetic masking it is all zeros). When a round key is first defined it's type is *original*. It may be masked or unmasked in this state, depending upon whether the values used to generate it were masked or not. When a specific mask is then used to further mask the round key it becomes the type *masked* (and is used to generate other new round keys). Finally when the round key is explicity masked for a second time its type becomes *final* (so that all round keys have the same final mask). In this model we assume that round keys are explicity masked at most two times (although this can be modified easily). The *type* of the round key has an ordering over time represented as $original \rightarrow masked \rightarrow final$. For example $t1 \rightarrow t2, t1, t2 \in type$, can be used to represent *t1,t2* as *original, masked* respectively or *t1,t2* as *masked, final*. The notation, $rk_i \oplus rk_j = rk_k$ represents the exclusive or of round keys i and j to produce round key k. The binary variable $x_{r,t,m}$ is one when the round key r of type t has the mask, m. The binary variable $mask_{r,t,m}$ is one when mask m is used to create r of type t. The binary variable $maskexists_m$ is one if the m is loaded from memory as a mask for one or more round keys. The binary variable $masked\_table_m$ is one if the masked table input mask is m, where $table_{m \oplus i} = masked\_table_i$, where table is the original unmasked table. The following parameters are used to describe the model: E_{load}, E_{opn}, represent the energy required to *load* a data word from memory and to perform an operation *opn*.

The energy overhead of the synthesized masking can be used as an objective function or in a constraint. Since each round key is assumed to be explicitly masked two times ($2n$) at the most, the second term of the energy function in (1) subtracts the number of unused masks (*mnull*) to obtain the number of additional operations used for masking. The first term in (1) is the number of masked used in total. The memory overhead can also be represented with the number of masks to be stored in memory.

$$\text{Energy overhead} = E_{load}\left(\sum_{m \in mask} maskexists_m\right) + E_{\oplus}\left(2n - \sum_{r \in rk, t \in type} mask_{r,t,"mnull"}\right) \quad (1)$$

$$\text{Memory overhead} = \left(\sum_{m \in mask} maskexists_m \times masksize_m\right)$$

The following constraints in (2) are used to ensure the binary variables assign one mask to each type of round key and the masked table has one input mask.

$$\sum_m x_{r,t,m} = 1, \forall r \in rk, t \in type$$
$$\sum_m mask_{r,t,m} = 1, \forall r \in rk, t \in type \mid type \neq original \quad (2)$$
$$\sum_m masked\_table_m = 1$$

Round keys can be masked or not masked within the algorithm at most two times. The round key masking inequality is first converted to propositional logic. For example the logical inference: if rk_i has an original mask *m1* (proposition *p1*) and rk_i has a resultant mask *m3* (*p3*) then the mask *m2* was applied (*p2*) (where $m1 \oplus m2 = m3$); can be represented by $(p1 \wedge p3) \rightarrow p2$. Next the propositional logic can be transformed into this disjunction $\neg(p1 \wedge p3) \vee p2$. This disjunction becomes $\neg p1 \vee \neg p3 \vee p2$. Since there is no more than one non-negated term, it is called a Horn clause. This disjunction be translated directly into an inequality, by replacing the first two negative propositions by one minus the binary variables, translating the $\vee$ symbol into an addition and setting the expression to greater than or equal to one. The inequality (3) illustrates the final form of this constraint.

$$1 - x_{i,t1,m1} + 1 - x_{i,t2,m3} + mask_{i,t2,m2} \geq 1, \quad \forall i \in rk, m1, m2, m3 \in mask \mid m1 \oplus m2 = m3, t1 \rightarrow t2 \quad (3)$$

The cryptographic operations which generate round keys are next used to formulate constraints for the model. For AES key generation, the round keys are generated from the exclusive or of other round keys. The unmasked round key generation from AES key scheduling, in general, is represented by $rk_i \oplus rk_j = rk_k$. When rk_i and rk_j are masked, the equation becomes $(m1 \oplus rk_i) \oplus (m2 \oplus rk_j) = (m3) \oplus rk_k$, for any m1, m2,

m3 where $m1 \oplus m2 = m3$. The round key generation inequality is derived from the logical inference : if rk_i has mask *m1* and rk_k has mask *m3* then rk_j must have mask *m2*. This Horn clause becomes transformed into the inequality (4). For example from the previous section, for i=0, this constraint is generated for round key operations *rk5=rk1* $\oplus$ *rk4* , by *4* $\oplus$ *5=1* (for $i \oplus j=k$ in (4)).

$$1 - x_{i,"masked",m1} + 1 - x_{k,"original",m3} + x_{j,"masked",m2} \geq 1, \quad (4)$$
$$\forall i, j, k \in rk, m1, m2, m3 \in mask \mid i \oplus j = k$$

The next inequality involves a second type of operation from AES key generation, where some round keys are generated from the exclusive or of a round key and a table look up. For example, $rk_4 = keyrotate(rk_3) \oplus rk_0 \oplus con(0)$, where the *keyrotate()* table has a masked output and *con(0)* is a constant. This round key generation for masking purposes will be represented as $rk(4) = table \oplus rk(0)$ in the model. Again the inequality is formed from logical inferences as given below in (5).

$$1 - x_{i,"masked",m1} + 1 - x_{k,"original",m3} + masked\_table_{m2} \geq 1, \quad (5)$$
$$\forall i, k \in rk, m1, m2, m3 \in mask \mid i \oplus table = k$$

The variable $maskexists_m$ can be defined from the variable $mask_{r,t,m}$ using the two generalized upper bound inequalities as given in (6).

In this masking optimization problem the final mask of all the round keys must be the same to avoid table regeneration. In AES, to reduce the number of variables, four masks *ma,mb,mc,md* and all combinations of them are considered in generating the set *mask*. The masking begins by masking the 128bit master key, represented as four 32bit (round keys) words *0, 1, 2, 3* with *ma,mb,mc,md*. The final mask on all round keys is set to $ma \oplus mb \oplus mc \oplus md$ (where $mabcd \in mask$ represents this mask). Respectively this can be represented by the following settings of constraints on variables in (7).

$$mask_{r,t,m} \leq maskexists_m,$$
$$\forall r \in rk, t \in type, m \in mask \mid m \neq mnull$$
$$\sum_{r \in rk, t \in type1} mask_{r,t,m} \geq maskexists_m, \quad (6)$$
$$\forall type1 \in \{masked, final\}, m \in mask \mid m \neq mnull$$

To ensure that the final mask is never computed (to avoid a 2nd order DPA attack) we add the following security constraint, (8). Unlike previous research[9], the masks of each round key are never computed (since they automatically result from masking before the key generation process), hence a higher security is achieved.

$$x_{0,"original",ma} = x_{1,"original",mb} = x_{2,"original",mc} = x_{3,"original",md} = 1, \quad (7)$$
$$maskexists_m = 1, \forall m \in \{ma, mb, mc, md\}$$
$$x_{r,"final",mabcd} = 1, \forall r \in rk$$
$$mask_{r,t,mabcd} = 0, \forall r \in rk, t \in type \quad (8)$$

The general model illustrated for AES key generation can also support loops. For example in AES, the mask of round key *0* must be the same as the mask for round key *8*, to support the loop in section 2. To support this loop, eight round keys (*0* to *7*) are used and inequality (4), for i $\oplus$ j=k, would be generated for 5 $\oplus$ 0=1 instead of *5* $\oplus$ *8=9*, 6 $\oplus$ 1=2 instead of 6 $\oplus$ 9=10, etc. Conditionals can also be supported (although these constructs are not advisable in security algorithms since their power profiles are easily identifiable, thus leaking information) by adding constraints that the variables must have the same mask after conditional blocks have been exited. The model can also support mask randomization (where random values are used to change hamming weights of pairs of masks in key generation). Randomization allows the individual mask values to change (thus thwarting hamming weight attacks), yet provides minimized energy overhead by keeping the final mask for round keys fixed. For example additional costs on certain masks would be added for extra energy overhead with randomization. As an example consider randomizing masks *ma* through *md* with random values *r1, r2*, where *ma* $\oplus$ *=r1, mb* $\oplus$ *=r1 and mc* $\oplus$ *=r2, md* $\oplus$ *=r2*. The final fixed mask input to the tables ($ma \oplus mb \oplus mc \oplus md$) does not change (since *r1* and *r2* cancel out), however many other masks will be randomized. For example, consider *mi*, where $mi=ma \oplus mc$, randomizing the masks will require that *mi* be updated by $r1 \oplus r2$, hence adding additional energy overhead which must be accounted for. Alternatively one can provide complete randomization of all masks and intermediate masks, using random values *rx, ry, rz, rw* (where $rw = rx \oplus ry \oplus rz$) , where *ma* $\oplus$ *=rx, mb* $\oplus$ *=ry and mc* $\oplus$ *=rz, md* $\oplus$ *=rw.*

Other key generation operations [13] such as rotation, permutation, shifting, addition, etc can also be supported in the model. This is facilitated through the use of mask sets where a general operation is represented by a mapping from one or two sets of masks into one other mask. For example in DES the subkey gets successively rotated by 1 or 2 bits in each round in DES. Since the rotated subkey at each round becomes joined and permuted in a fixed way to become the round key, the optimization for masking involves the subkey (which directly transforms into a round key). Depending upon the memory requirements of the application the user can define different sized mask sets to be used in the optimization algorithm. Specifically the mask set can be initiated with an initial mask on the first subkey, *m*. For example by supporting masks of at most 2bit rotations, one would support masks: $m, m \lhd 1, m \lhd 2, ((m \lhd 1) \oplus m), ((m \lhd 2) \oplus m),$ $((m \lhd 1) \oplus (m \lhd 2)), ((m \lhd 1) \oplus (m \lhd 2) \oplus m)$, or as represented in the model as *m0,m1,m2,m3,m4,m5,m6* respectively, where $\lhd w$ represents rotation to the left by *w*). For example an inequality supporting masking of a subkey or round key, *i*, (where *srk* is the set of round keys and subkeys) could be represented in (9).

$$1 - x_{i,t1,m3} + 1 - x_{i,t2,m0} + mask_{i,t2,m1} \geq 1,$$
$$\forall i, \in srk, m1, m0, m3 \in mask \quad . \quad (9)$$
$$\mid m3 \oplus m1 = m0, t1 \rightarrow t2$$

For example since the subkey is successively rotated, a larger mask set may support rotations up to *x* bits, *x>2*. In this example a final fixed mask could be chosen as any of *m2* through *m7*. The model can also be applied to arithmetic masking where instead of the exclusive or operation, one would use the modulo addition operation.

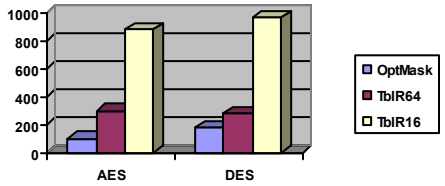
4. EXPERIMENTAL RESULTS

The ILP optimization model results, the energy overheads, and evaluation of the low energy countermeasure security will be presented in this section. The optimization model was solved using IBM's OSL optimization software on a PC laptop. Real power measurements were used to evaluate the security and energy of the optimization technique. A high sample rate oscilloscope, a trigger probe, a differential probe and an ARM7TDMI evaluation board (providing access to the core's power supply) were used to acquire power traces. The differential 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 trigger signal was controlled by software and measured with a probe in order to synchronize the measurements. The scope sampled at rates up to 2.5Gsamples/sec, and allowed many power traces to be acquired automatically. The evaluation board 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.

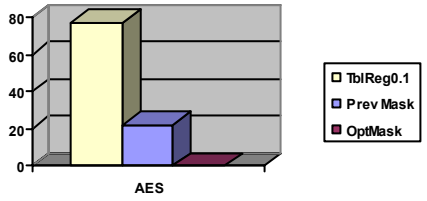
The ILP optimization model was created for AES and DES. In AES, loops were supported and masking on the 8 round keys was optimized for energy. The model had 673 variables and 7.9K equations, requiring 23 nodes in the branch and bound tree to minimize the energy overhead with given memory constraints. Another model for masking with 16 round keys had 1313 variables and 15.8K equations and solved the optimization with only 19 nodes searched in the branch and bound tree. The small number of nodes in the branch and bound tree is most likely due to the Horn clauses and logic inference structure in the ILP similar to previous research findings[12]. In AES the minimum energy solution utilized 9 masks and the minimum memory sized solution used 6 masks.

Figure 1 illustrates the energy overhead of the previous research where table regeneration is required compared to the optimized approach presented in this paper. The energy required to regenerate the tables in AES is an order of magnitude larger than the energy required to encrypt 128bits of data. In DES the table regeneration energy is approximately 5 times larger than encryption 64bits of data in DES. The total energy for key generation and encryption of 2KB using AES and DES is shown in figure 1a) due to the optimization approach (OptMask), and previous research requiring table regeneration after every 64 (TblR64) and 16 (TblR16) bytes of data being encrypted. Figure 1b) illustrates the energy overhead required by the countermeasures during encryption with table regeneration (scaled by 0.1), previous masking and optimized masking for 2KB of data. For 128bits of data, the energy savings is 2.5 times over previous masking approaches where masking is only performed within the encryption (without accounting for table regeneration). In comparison with [9] which requires at least double the number of tables, our approach also requires less energy since a smaller memory could be used. For example, one set of tables is required along with storage of 20 words, for 4 masks (*ma,mb,mc,md*) and 16 intermediate masks (for the masking solution of 16 round keys within a loop).

To evaluate the security of the low energy optimized masking countermeasure, both hamming weight attacks (for non-randomized implementation) and DPA attacks were analyzed. Although our real power measurements were unable to correlate the power to hamming weights, assuming it may be possible with other processors, the security was evaluated. Fig 2a) illustrates the number of possible key guesses which could be derived from the hamming weight of the key, mask and the exclusive or, $key \oplus mask$. Figure 2a) illustrates a hamming weight attack possible on previously researched masking. The optimized approach uses multiple masks, hence the number of key guesses



a)

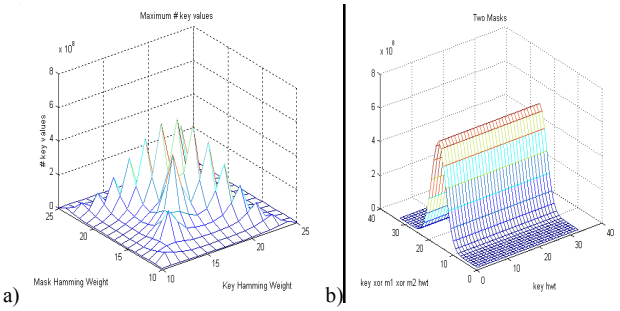


b)

Fig 1. a) Energy comparison (uJ) for 2KB data encryption using previous researched techniques and the optimized approach (OptMask). Energy Overheads in b).

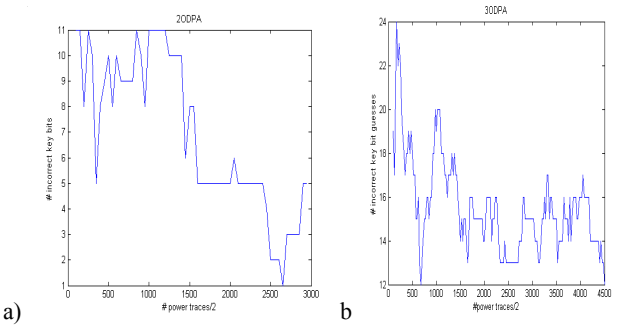
possible given the hamming weight of the first mask (*m1*), the second mask (*m2*), the key and the exclusive or of all three ($m1 \oplus m2 \oplus key$) is illustrated in Fig 2b). In both cases the maximum number of possible solutions is 601 million solutions (for a number of hamming weight combinations). The average number and sum of key guesses increased by 8 and 280 times respectively in b) over a). Hence security has increased for the low energy split masks countermeasure optimization.

The second order (from previous research) and third order (from optimized approach) DPA results were obtained using real power measurements. A third order DPA (requiring three power samples) is required for this optimized split mask approach since power samples of mask $ma \oplus mb$, mask $mc \oplus md$, and the exclusive or results of the masked round key (whose final mask is *mabcd*) and the plaintext are necessary. A second order DPA can only be supported if the mask *mabcd* was computed (using two power sample : mask *mabcd* and the exclusive or result) and in our optimized approach the security constraint ensures it never is computed. The results are shown in figure 3a) and b) respectively. In figure 3a), after 2500 power traces, the 2nd order DPA incorrectly predicts only 1 key bit out of 32 bits. However in Fig 3b) (the third order DPA) after analysis with 4500 power traces, 12 key bits were incorrectly predicted out of 32 bits. These results also support the increased security from using optimized split masks.



a) b)

Fig. 2. Number of key guesses for one a), and two split masks b).



a) b

Fig. 3. The 2nd a), and 3rd b) order DPA results using real power.

5. DISCUSSION AND CONCLUSIONS

This paper presented for the first time an optimization based methodology using split masks and masking before key generation to provide a low energy countermeasure for embedded cryptographic applications. However for embedded systems where energy is not as important as security, this optimized methodology can also be used with table regeneration, such that new random values of *ma,mb,mc,md* are generated for each AES task or round. For the low energy countermeasure, where tables are not regenerated, added security (requiring no less than a 3rd order DPA) is provided through the split masks. Hamming weight attacks are also investigated, and found to add security over a single mask case. Furthermore real power results illustrate the difficulty of obtaining the key values from a 3rd order DPA. Hence the security has been assessed along with the energy overhead savings to support this optimization methodology as being useful for many embedded energy constrained systems. Results with real power measurements for the ARM7TDMI processor core showed that finding all bits of the secret key required more than 4500 power traces (unlike previous research which investigated only a few bits on an 8 bit processor[10]), indicating that this low-energy high-performance countermeasure is very powerful and suitable for embedded or wireless devices. The countermeasure was focused on the key XORing attack previously studied[10]. However other attacks, such as the Sbox DPA attacks exist, and low energy countermeasures for these should also be incorporated.

This study presents a methodology for optimized masking and for minimizing the memory and energy overheads of power analysis countermeasures for cryptographic applications. Unlike previous research [9,10], an optimization approach has been developed to aid countermeasure synthesis within an application. The optimization model uses a new concept called split masks, which unlike previous research which splits the key, this approach splits the mask of the key into multiple masks enforcing a higher order DPA. Real power measurements have confirmed the higher resistance to DPA and hamming weight attacks. Most important the reduction in energy dissipation is significant compared to previously researched approaches and can be optimized within a model supporting memory and security constraints. This research is crucial for supporting low energy and high performance security for embedded systems which will be prevalent in wireless and embedded devices designed with nanometer technologies of the future. The author thanks S.Bulgin for his work and financial support from NSERC, CITO, and RIM.

REFERENCES

[1] P.Kocher, J.Jaffe, B.Jun "Differential Power Analysis" Crypto'99, LNCS 1666, 1999

[2] S.Ravi, etal. "Securing Wireless Data: System architecture challenges", ISSS, 2002, pp.195-200.

[3] J.Coron, etal. "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 AES finalists against power analysis attacks" LNCS 1978, 2001, pp.150-164.

[6] 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.

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

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

[9] K.Itoh etal. "DPA countermeasure based on the masking method", LNCS 2288, 2002,pp.440-456.

[10] T.Messerges "Using 2nd Order Power Analysis to attack DPA resistant software", LNCS 1965, 2000 pp.238-251.

[11] Chandrasekharan "Integer Programming problems..works", Progress in Comb.Opt. 1984, pp.101-6.

[12] Hooker "Resolution vs Cutting plane ..problems" Oper. Res. Letters Vol.7,No.1(1), 1988.

[13] B. Schneier, Applied Cryptography, J.Wiley, 1996.

[14] J.Golic "Multiplicative Masking and power analysis of AES", CHES 2002.

[15] http://csr.nist.gov/encryption/aes/rijndael/Rijndael.pdf.