DRECON: DPA Resistant Encryption by Construction

Suvadeep Hajra[1], Chester Rebeiro[1], Shivam Bhasin[2], Gaurav Bajaj[1], Sahil Sharma[1], Sylvain Guilley[2,3], and Debdeep Mukhopadhyay[1]

[1] Dept. of Computer Science and Engineering,
Indian Institute of Technology Kharagpur, India
{suvadeep.hajra,chetrebeiro,bajaj.gaurav92,shlshrm000,
debdeep.mukhopadhyay}@gmail.com
[2] Institut MINES-TELECOM, TELECOM ParisTech,
Department COMELEC, 46 rue Barrault,
75634 PARIS Cedex 13, France
{bhasin,guilley}@telecom-paristech.fr
[3] Secure-IC S.A.S., 80 avenue des Buttes de Coësmes,
35700 Rennes, France

Abstract. Side-channel attacks are considered as one of the biggest threats against modern crypto-systems. This motivates the design of ciphers which are naturally resistant against side-channel attacks. The present paper proposes a scheme called DRECON to construct a block cipher with innate protection against differential power attacks (DPA). The scheme is motivated by tweakable block ciphers and is shown to be secure against first-order DPA using information theoretic metrics. DRECON is shown to be less expensive than masking and re-keying countermeasures from the implementation perspective and can be efficiently realized in both hardware and software platforms. On FPGAs especially, DRECON can optimally utilize the abundant block RAMs available and therefore have minimal overheads. We estimate the cost overhead of DRECON in micro-controllers and FPGAs, two common targets for cryptographic applications. Finally we demonstrate practical side-channel resistance of a DRECON implementation on a Xilinx Virtex-5 FPGA (SASEBO GII board).

1 Introduction

In 1998, Paul Kocher demonstrated a new class of cryptographic attacks known as differential power analysis (DPA) [13], which utilize information leakages from power or electro-magnetic radiation of the cipher's implementation. Since then, several DPA attacks have been demonstrated on almost every crypto-system in use. Today DPA has become one of the biggest threats to modern security systems. Over the years there have been several attempts to prevent these attacks. A current trend is to either eliminate [37,38] or randomize [2,7] side-channel leakage. An alternate trend is a modification of the protocols, for example,

D. Pointcheval and D. Vergnaud (Eds.): AFRICACRYPT 2014, LNCS 8469, pp. 420–439, 2014.

by changing a secret in every encryption [8,19,28]. However in both these prevention methodologies, the underlying cryptographic algorithm is unchanged and by itself remains weak against the attack. Further, many of the countermeasures are ad-hoc, platform dependent, or require customized development processes. It has also been seen that several countermeasures can only make the attack more difficult and not fully protected.

Better protection can be achieved if the cryptographic algorithm itself is designed with DPA in the hindsight in addition to the conventional cryptanalytic attacks as the primitives used can be chosen with better side-channel attack resistance. Due to this reason, research is being carried out in developing cryptographic primitives that are easily protectable by masking [29,6], or that can inherently tolerate these attacks. In this paper, we show how cryptographically good sboxes can be arranged in such a way that would result in a cipher with increased resistance against DPA. This strategy ensures DPA resistance without compromising on classical cryptographic properties.

The scheme we present is called DRECON (DPA Resistant Encryption by CONstruction), which attempts to design a complete block cipher with DPA prevention as a pre-requisite. The scheme currently guarantees first-order security, and can be used as a starting point to build ciphers with higher order DPA resistance. The construction is inspired from tweakable block ciphers [14], where in addition to the plaintext and key, the cipher takes a *tweak*. However, unlike the tweakable block ciphers in [14], the construction requires the tweak to be kept secret. The tweak is used to choose an sbox from a given pool of cryptographically strong sboxes, thus modifying the mapping between the plaintext and the ciphertext. Protection against DPA is obtained based on the assumption that the tweak is exclusively shared between the sender and the receiver and modified in every encryption. Besides the fact that the primitives used in DRECON have higher resistance against DPA attacks, there are several advantages over contemporary DPA countermeasures. Compared to randomization techniques such as masking, we show that encryptions in software can be done faster. In hardware, the area and performance overheads are considerably less compared to masking. Further no custom libraries or design flows are required as compared to hiding countermeasures such as [37,38] and unlike protocol countermeasures such as [8,19,28], key expansion needs to be done just once. The construction is supported by information theoretic proofs of security. We show that the DRECON is resistant against the first-order (1O) DPA attack in the presence of glitches also. We have experimentally validated the result on a version of DRECON using the powerful correlation-collision attacks.

The organization of the paper is as follows: in Section 2, the necessary background for DPA is presented along with an introduction to commonly used countermeasures. Section 3 presents DRECON and evaluates its security against DPA. In Section 4, implementation aspects of the scheme are presented for both hardware and software platforms. We also validate its resistance to correlation-collision attack on the SASEBO-GII side-channel evaluation board [33]. The final section has the conclusion of the paper.

2 Preliminaries

2.1 Notations

We denote random variables by capital letters (e.g. X) and their realization by small letters (e.g. x). The universe for the variable is represented as a calligraphic letter (e.g. $\mathcal{X}$).

Let x be part of the plaintext which gets ex-ored with part of a sub-key, k, in an encrypting block cipher. Assume that x and k are chosen from $\mathcal{X}$ and $\mathcal{K}$ respectively and its choice is represented by the random variable X and K respectively. Generally, X is ex-ored with a unknown but fixed key k and then undergoes a non-linear transformation with an sbox. We denote this operation by $S(X \oplus k)$.

2.2 Differential Power Attacks

The aim of a DPA adversary is to use the side-channel leakage from either $X \oplus k$ or $S(X \oplus k)$ to reveal the secret data k. The steps involved is to rank each possible candidate $k^* \in \mathcal{K}$ for the key k by statistically comparing the actual leakage with a model of the leakage. The candidate ranked first is the most likely and the one ranked last is the least likely candidate. The $o-$th order average success rate of the attack is the probability with which the correct value of k has a rank between 1 and o [36].

The success of the attack depends on how much information gets leaked. In 2004, Micali and Reyzin used leakage functions (denoted ϕ) to encapsulate the information leaked through the side-channels [21]. The *ideal leakage function* is one which leaks the entire internal state of the cipher. It can be defined as:

$$\mathsf{Id} : y \in \mathbb{F}_2^n \mapsto y \in \mathbb{F}_2^n \ , \tag{1}$$

where $y = S(x \oplus k)$ is the intermediate state. A more realistic leakage function is the Hamming weight leakage, which leaks the Hamming weight of y. It is defined as

$$\mathsf{HW} : y \in \mathbb{F}_2^n \mapsto \sum_{i=0}^{n-1} y_i \in \mathbb{N} \ , \tag{2}$$

where y_i is the i^{th} bit of the intermediate state y.

2.3 Countermeasures for DPA

Countermeasures for DPA are applied at the implementation level or at the protocol level. The most common countermeasures used in the implementation level are masking, shuffling, and hiding. The advantage of this is that they can be applied on any cipher algorithm. On the other hand, they are affected by the platform of implementation and do not always provide provable security. Several *hiding* schemes have been proposed, which essentially use side-channel resistant logic styles in order to prevent information leakage through the power consumption. Examples of this can be found in [37,38]. These countermeasures may require specific CMOS libraries or full custom designs. Masking and shuffling do not have these limitations and will be discussed here in greater detail.

Masking: *Masking* is the most frequently used countermeasure [2,7] applied to implementations. A p-order masking scheme involves spreading each sensitive variable Z into $p+1$ shares $Z_0, \ldots, Z_p$ maintaining the invariant $Z = g(Z_0, \cdots, Z_p)$. Each of the Z_i's are uniformly random and the joint distribution of any p variables are independent of Z. Thus, any collection of variables less than or equal to p contains no information about the sensitive variable Z. The most commonly used masking is the *first order masking* (denoted $1\mathsf{O}$ masking) where a single uniformly random mask is used.

Let M be a random variable M with entropy $h_m \leq n$. In the $1\mathsf{O}$ Boolean masking scheme, it gets added to the sensitive variable $X \oplus k$ resulting in two shares: $X \oplus k \oplus M$ and M. Each sbox S is also replaced by a masked sbox S_M such that $S_M(X \oplus k \oplus M, M, M') = S(X \oplus k) \oplus M'$. In other words, the masked sbox S_M first unmasks the randomized variable $X \oplus k \oplus M$, passes it through the sbox S and then re-masks the output $S(X \oplus k)$ by the output mask M'. The mask M is also replaced by the new mask M', thus the invariant $S(X \oplus k) = S_M(X \oplus k \oplus M, M, M') \oplus M'$ is maintained. Now, the $1\mathsf{O}$ side-channel leakage has the form $\phi(S(X \oplus k) \oplus M')$. In the case where the leakage function $\phi = \mathsf{Id}$ (Equation 1), the entire output $S(X \oplus k) \oplus M'$ is revealed to the adversary. Information leakage is measured by the mutual information (abridged I) between what can be observed by the attacker and the sensitive variable. Renaming the variable M' as M:

$$\mathsf{I}[S(X \oplus K) \oplus M, X; K] = n - h_m \ . \tag{3}$$

This means that the countermeasure is perfect at $1\mathsf{O}$ if and only if M has entropy $h_m = n$, *i.e.* M is uniformly distributed over $\mathbb{F}_2^n$.

In the case where $\phi = \mathsf{HW}$ (defined in Equation 2), only the Hamming weight of $S(X \oplus k) \oplus M$ is revealed. The information leakage is equal to:

$$\mathsf{I}[\mathsf{HW}(S(X \oplus K) \oplus M), X; K] = \mathsf{H}[\mathsf{HW}(K)] - \sum_{x,k} \mathsf{P}[K = k]\mathsf{P}[X = x] \cdot \mathsf{H}[\mathsf{HW}(S(x \oplus k) \oplus M)] \ . \tag{4}$$

If M is independent of X, the second term of the difference is equals to $\mathsf{H}[\mathsf{HW}(M)]$. The value of Equation (4) is lower than that of Equation (3), but a priori hard to make null if $h_m < n$.

If $h_m = n$, the single mask can perfectly shield against first-order attacks but not against attacks of higher order such as [20]. This is because, in a second-order attack the adversary is capable of obtaining the leakage $\phi(M)$ in addition to $\phi(S(X \oplus k) \oplus M)$. However, the complexity of the attack increases. The complexity is reduced significantly when the computations of M and $S(X \oplus k) \oplus M$ overlap. The leakage then takes the form $\phi(M) + \phi(S(X \oplus k) \oplus M)$ and follows a distribution whose higher order moment depends on $X \oplus k$. These attacks are known as *univariate higher order attacks* (the one which uses variance is called univariate second order attack [39]).

A $1\mathsf{O}$ masking scheme is secure against a $1\mathsf{O}$ attack in an idealistic model. Most of the model is based on the assumption that the output of a circuit switches only once in a clock cycle. However due to asymmetric path delay, output of

a CMOS gate may switch more than once in a clock cycle. This phenomenon is referred to as 'glitch' [16]. Since most of the masking schemes combine the masked value $X \oplus k \oplus M$ and the masks M, M' within the same combinatorial circuit, the leakage takes the form $\phi(X \oplus k \oplus M, M, M')$. Due to the glitches, this leakage $\phi(X \oplus k \oplus M, M, M')$ becomes strongly correlated to the unmasked sensitive variable $X \oplus k$ [27]. Consequently, the circuit becomes vulnerable to first order DPA attacks [17,4,22,23]. Secure implementation of non-linear function in the presence of glitches has been proposed in [27]. However, implementation of such a scheme increases the hardware cost drastically [25,31].

Shuffling: An alternate randomization technique is *shuffling* [10,34]. Here instead of a random mask being added, executions of several sensitive operations are shuffled in time. If the execution of an operation is spread over m different signals, then the information per signal is reduced m times. This works well because DPA can target a single signal at a time. However 1O DPA attacks can defeat shuffling using m^2 times traces [3].

Protocol Level: DPA requires several power traces in order to successfully retrieve the secret key due to the noise present in the target device and due to the non-injective nature of the leakage function. Protocol level countermeasures prevent the adversary from collecting the required number of traces. In [28], Kocher suggests to update the key on a regular basis. The rate of updation should be fast enough to prevent an adversary from collecting the necessary traces. The updation rate should be evaluated for each device and cipher implementation. In the strongest form, every encryption is done with a new key.

There are various ways in which key updation (or *re-keying*) can be done. Abdalla and Bellare in [1] classify them into two schemes: parallel and serial. In the *parallel re-keying scheme*, a key update is derived directly from the master key using a suitable function f. For example the i^{th} key update (denoted K_i) can be obtained from $K_i = f(K, i)$, where K is a master key. Methods of key updation using this scheme have been suggested in [19] and [8]. To obtain provable security using the scheme, it is required that the key updates are precomputed and stored in memory. Thus the number of encryptions is limited by the size of memory. In the serial re-keying scheme, a new key is obtained from the previous key using a suitable function f. For example, the i^{th} key can be obtained from the previous key as follows: $K_i = f(K_{i-1})$, while the first key used is derived from the master key (*i.e.* $K_1 = f(K)$). Re-keying mechanisms using this technique were suggested in [28] and [18].

The drawback of the re-keying mechanisms is that with each key update, new round keys have to be computed. Thus, the overhead is not only in the generation of the new key, but also the computation of the key expansion algorithm. This can add significant overheads in the performance, especially in software implementations. Our proposal does not suffer from this drawback. In DRECON, the round keys are fixed for all encryptions. Instead, only the tweak is updated by a function similar to f used in the previous schemes.

3 The DRECON Scheme

In 2002, Liskov, Rivest, and Wagner introduced *tweakable block ciphers* to add more variability to the functionality [14]. Here an additional secret input called the *tweak* is present, which if changed alters the map between the plaintext and ciphertext thereby obtaining more variations in the mapping. Both the sender and the receiver need to know the tweak in addition to the secret key. The proposal in this paper is inspired by tweakable block ciphers, and uses a regularly changing tweak to stymie differential power attacks. In this section, we present the proposal and then compare its security with that of 1O masking.

DRECON: The secret in DRECON comprises of the tuple (t, k), where t is called the tweak and k the key used in the block cipher. The key k is held constant for all encryptions, while the tweak t changes for each encryption, using a tweak generation algorithm. The tweak is used to select a function from the set $\mathcal{F}\{\mathsf{F}_1, \mathsf{F}_2, \cdots, \mathsf{F}_r\}$, where $\mathsf{F}_j : \{0,1\}^n \mapsto \{0,1\}^n$ and $(1 \leq j \leq r)$, are cryptographically strong sbox functions. For every application of the sbox on X, a function from $\mathcal{F}$ is selected depending on the value of the tweak (t) and applied to X. This sbox, known as the *tweaked-sbox*, is represented by $\boldsymbol{S}(\cdot, \cdot)$ and defined as follows:

$$\boldsymbol{S}(t, X) \leftarrow \mathsf{F}_t(X) \qquad \text{where } t \overset{R}{\leftarrow} \{1, 2, \cdots, r\}.$$

In a typical iterative block cipher, the first round key is added to plaintext before the sbox operation and the sbox operation has the form $S(x \oplus k)$. However, in DRECON, we choose to omit the whitening at the beginning of the encryption. Thus, each round except the last round consists of substitution layer, diffusion layer and key addition layer. The last round consists of only substitution layer. The sboxes of the substitution layers is replaced by the tweaked-sbox. For all round, the same tweaks are used though two different tweaked-sboxes of the same round use two different tweaks independently. The first round of DRECON is shown in Figure 1. It may be noted that DRECON requires no key whitening at the beginning and end of the block cipher since the tweaked-sboxes provide the required randomization of the input and output respectively.

3.1 Information Theoretic Analysis

First we will analyse the security of the above scheme for glitch-free circuit. Then we will analyse its resistance in the presence of glitches. For all the analysis, we consider the known plaintext attacks where its distribution is uniformly random.

In the Absence of Glitch: Let us assume there is no diffusion layer, since its presence does not dilute the side-channel security of DRECON. Let T be the random variable representing the tweak and having entropy h_t. In the worst case, the entire state gets leaked (that is $\phi = \mathsf{Id}$). Hence, one can get the 1O leakage $\boldsymbol{S}(T, X) \oplus k$. The information leakage can be shown to be

$$\mathsf{I}[\boldsymbol{S}(T, X) \oplus K, X; K] = \mathsf{H}[X] - \sum_x \mathsf{P}[X = x] \cdot \mathsf{H}[\boldsymbol{S}(T, x)] \ . \tag{5}$$

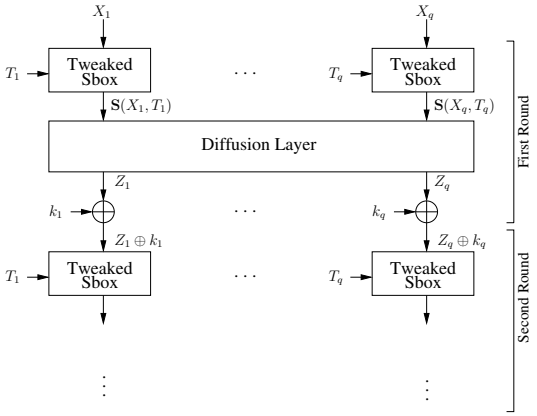


Fig. 1. First round of DRECON. The same structure is repeated for all rounds except the last round which is consisted of only substitution layer.

This quantity is greater than or equal to $n - h_t$ if $h_t \leq n$. When $h_t > n$, it is greater than or equal to 0.

Comparing Equations (5) and (3), we note that DRECON performs as good as $\mathsf{1O}$ masking scheme for the Id leakage functions (the worst case), provided the following propositions are respected.

Proposition 1. *$\mathsf{I}[\boldsymbol{S}(T, X) \oplus K, X; K] = n - h_t$ if and only if $\nexists (x, t_0, t_1) \in \mathbb{F}_2^n \times \mathcal{T} \times \mathcal{T}$, such that $\boldsymbol{S}(t_0, x) = \boldsymbol{S}(t_1, x)$.*

This means that for all $x \in \mathbb{F}_2^n$, the values taken by the random variable $\boldsymbol{S}(T, x)$ are of cardinality 2^{h_t}.

Proposition 2. *$\mathsf{I}[\boldsymbol{S}(T, X) \oplus K, X; K] = n - h_t$ if and only if $\forall x, t \in \mathcal{T} \mapsto \boldsymbol{S}(t, x) \in \mathbb{F}_2^n$ is balanced.*

Thus, to make the information leakage null, h_t should be atleast n. In other words, the tweak should be atleast n bit long.

In the Presence of Glitches: Let us first assume that there is no glitch in the key addition layer. This assumption is aligned with the existing observations in the literature [16] and much of the effort has been directed to make a non-linear circuit resistant in the presence of glitches [27]. We also assume that tweak T is following an uniformly random distribution with entropy $h_t \geq n$ and the tweaked-sbox satisfies the balancedness property of Proposition 2.

Under the above assumptions, the output of the substitution layer in the first round (see Figure 1) is a uniformly random unknown variable. Hence the leakages of the linear layer of the first round take the form $\phi(R \oplus k)$ for some uniformly distributed unknown random variable R. Thus, univariate attack targeting that layer is not feasible. However, the leakage of an sbox $\boldsymbol{S}(R \oplus k, T)$ in an intermediate round takes the form $\phi(R \oplus k, T)$ which may leak some information of

the secret k if there exists a dependency between R and T. To analyse the case further, we assume that the diffusion layer consists of q/m maximum distance separable (MDS) mapping $\mathsf{M} : (\{0,1\}^n)^m \mapsto (\{0,1\}^n)^m$ where q is the number of sboxes present in a single round, $m \geq 2$ and q is divisible by m. Lemma 1 provides the following result.

Lemma 1. *Let $X_1, \ldots, X_q$ be the random variables representing the plaintext inputs of DRECON. Let $Z \oplus k$ and T be the inputs to an sbox $\boldsymbol{S}(Z \oplus k, T)$ of an intermediate round where T is the random variable representing the tweak input to the sbox. Then the random variable Z is independent to the joint distribution of $X_1, \ldots, X_q$ and T.*

The proof of the above lemma is given in Appendix A. Before computing the bound of the information leakage, we state without proof two well known results of information theory [5]:

Lemma 2. *Let $U_1, \ldots, U_r$ be r mutually independent variables. Then*

$$\mathsf{H}[U_1, \ldots, U_r] = \sum_{i=1}^{r} \mathsf{H}[U_i]$$

Lemma 3. *Let U_1 and U_2 be two random variables. Then*

$$\mathsf{I}[U_1; U_2] = \mathsf{H}[U_1] + \mathsf{H}[U_2] - \mathsf{H}[U_1, U_2]$$

Applying the above three lemmas, the information leakage $\phi(Z \oplus k, T)$ due to the sbox $\boldsymbol{S}(Z \oplus k, T)$ can be computed as

$$\begin{aligned} \mathsf{I}[\phi(Z_1 \oplus K, T), X_1, \ldots, X_q; K] \leq & \mathsf{I}[Z_1 \oplus K, T, X_1, \ldots, X_q; K] \\ = & 0 \ . \end{aligned}$$

Hence, DRECON is resistant against $1\mathsf{O}$ DPA in the presence of glitches also.

DRECON and Shuffling: DRECON is not a shuffling countermeasure. A typical shuffling countermeasure would have computed the correct result sometimes and at others something which is totally uncorrelated from the data. Thus shuffling is not perfect at first order. On the other hand, DRECON is sound at $1\mathsf{O}$ using the mutual information metric.

DRECON and Masking: As discussed in Section 2.3, a simple masking like Boolean masking provides perfect secrecy against $1\mathsf{O}$ DPA only in glitch free circuits. By custom design of circuits, glitches can be reduced, but can never be eliminated totally. On the other hand, masking schemes that of [27] provides high resistance against $1\mathsf{O}$ DPA in the presence of glitches, but are very costly to implement. Thus, DRECON provides a cost-effective alternative to those masking schemes against $1\mathsf{O}$ DPA in the presence of glitches.

DRECON and Re-keying: DRECON, in some sense, is similar to re-keying mechanisms. However unlike re-keying, the key is held constant for all encryptions while it is the tweak that changes and needs to be synchronized between the sender and receiver. If the key is changed such as in [28,8,18,19], then in addition to the generation of the new key, the key expansion algorithm has to be executed in order to generate the new round keys. DRECON doesn't suffer from this drawback since it is only the tweak that needs to be generated.

4 An Application of the DRECON Scheme

DRECON is implementation friendly for both software and hardware platforms and can be easily derived from any legacy block cipher, preferably one which has small sboxes of dimensions for example 4×4. When implemented with DRECON, each of the sboxes of the legacy cipher is chosen from a pool of cryptographically equal strong sboxes for each encryption based on the unknown tweak such that Proposition 2 is satisfied. Thus, the classical blackbox cryptanalytic attacks are no more applicable under the assumption that the tweak is a uniformly random variable parameter.

In this article, we consider two implementations of DRECON which are based on AES algorithm. The first implementation is more resource friendly and based on a simplified AES algorithm. The simplified AES algorithm, which we name as 4×4 AES, follows the standard AES specification, except that the 8×8 sbox is replaced by a pair of 4×4 sboxes. Thus, there are 32 sbox access per round instead of 16 for the regular AES algorithm. The second implementation is based on the standard AES which we refer as 8×8 AES.

The adapted $n \times n$ AES algorithm with DRECON is called $n \times n$ DRECON-AES where possible values of n is 4 and 8. The DRECON-AES has the following properties. Each round of DRECON-AES has the same structure as that of $n \times n$ AES except the *AddRoundKeys* of the first and the last round are omitted. The *ShiftRow* operation of the last round is also omitted, and thus last round is left with only the *SubBytes* operation (Figure 2). Further, each $n \times n$ bit sbox is replaced by a $n \times n$ bit tweaked-sbox. Each tweaked-sbox is a set of 2^n ($r = 2^n, n = 4$ or 8) non-linear functions having the equal cryptographic strength, which put together satisfies Proposition 2. The criteria for selection of these sboxes is specified in Section 4.3.

4.1 Operation of DRECON-AES

Using DRECON-AES to secure communication between a sender and receiver has three phases as shown in Figure 3. The phases are explained below.

- **Bootstrapping:** To bootstrap, both parties need to agree on a secret key as well as a secret master tweak. Standard key exchange protocols can be used for the purpose.
- **Key Expansion:** The next step is to generate the round keys at both ends using a key scheduling algorithm. The sboxes in AES key scheduling algorithm are replaced by the tweaked-sbox. The tweak bits are generated by a

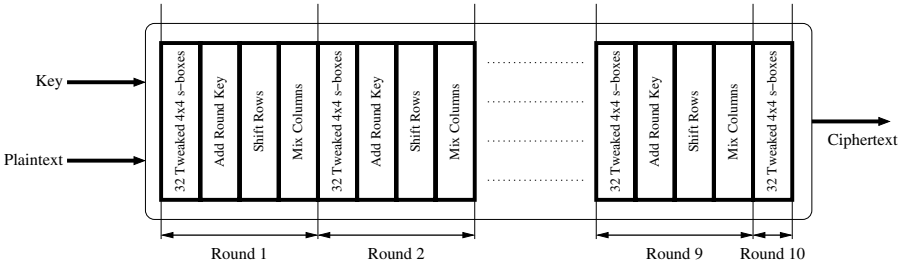


Fig. 2. 4×4 AES Adapted for DRECON

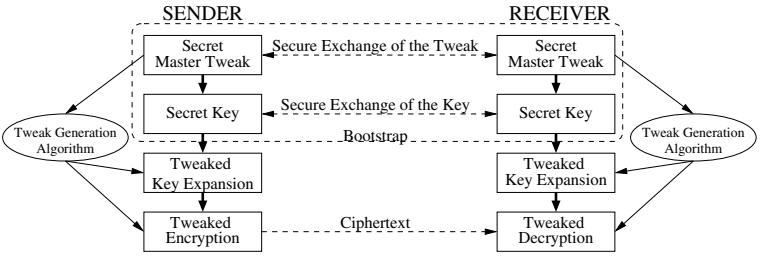


Fig. 3. Application of DRECON

tweak generation algorithm discussed in Section 4.2. The same tweak is used for all accesses during the key generation. The round keys are thus generated once and stored which are then used for every encryption until there is a change in the session key.

– **Encryption:** is then performed. Each encryption requires 128 bits of tweak to be generated, since each sbox takes a tweak input of equal size of its original input and all rounds use the same tweak.

The entire operation of the 4×4 DRECON-AES is summarized in Algorithm 1 of Appendix B. The operations of the 8×8 DRECON-AES are similar to those of 4×4 DRECON-AES except every pair of consecutive 4×4 tweaked-sboxes is replaced by a 8×8 tweaked-sbox.

4.2 Tweak Generation Algorithm

From the master tweak agreed upon by the sender and receiver, tweaks need to be generated for each encryption. The tweak generation needs to produce uniformly random tweaks in the range of 1 to r in order to select one of the r sboxes (for DRECON-AES $r = 16$ or 256). Further, the algorithm needs to be secure against power attacks as is discussed in detail in [19].

Any mask generation function (MGF) or stream cipher implemented in a secure manner can be used as a tweak generator. However, given the fact that the adversary has no control or knowledge of the input and output of the tweak generator, lightweight solutions can be developed by balancing registers and

minimizing the combinational logic, which can otherwise leak [9]. A possible construction for a tweak generation algorithm makes use of an LFSR as shown in figure 4. The design uses two pairs of shift registers (S and $\overline{S}$), each comprising of n flip-flops. The flip-flops in $\overline{S}$ are a complement of the flip-flops in S. To obtain such a state, the master tweak is used to seed S and the complement of the master tweak is used to seed $\overline{S}$. Further, the feedback obtained from an n degree primitive polynomial is complemented before being fed back to $\overline{S}$. Since all clocks toggle at the same time, the leakage from the registers is minimised. The alternate source of leakage, from the combinational paths, is also kept minimum by choosing a primitive polynomial with small number of coefficients. For DRECON-AES, $n = 128$ and the primitive polynomial chosen was $\alpha^{128} \oplus \alpha^7 \oplus \alpha^2 \oplus \alpha \oplus 1$.

4.3 Choosing the S-boxes

Proposition 2 mandates that in order to make $\mathsf{I}[\boldsymbol{S}(T, X) \oplus K, X; K]$ minimum, $\boldsymbol{S}(T, x)$ should be balanced for all x. To make $\mathsf{I}[\boldsymbol{S}(T, X) \oplus K, X; K]$ zero, it is enough to have $h_t = n$. Again to make $\boldsymbol{S}(T, x)$ balanced, we can choose the size of $\mathcal{T}$ to be 2^n. Further, each of the sboxes needs to have good cryptographic properties to ensure security against black box attacks.

Exhaustive search can be used to find such sboxes. However, when the size of the sbox is large, it becomes infeasible. We choose the set of sboxes which are obtained using an affine transformations of a cryptographically strong sbox. That is, if $S(\cdot)$ is a cryptographically strong sbox, we find a set of 2^n strong sboxes by setting $\mathsf{F}_i(x) = \alpha \mathsf{S}(x) \oplus i$ for all $i = 0, \cdots, 2^n - 1$ where α is an invertible matrix of dimensions $n \times n$. Since affine transformation does not changes the cryptographic properties of sboxes, all the sboxes of the set possess equal cryptographic straingth of the original sbox.

4.4 Software Implementation of DRECON-AES

DRECON-AES can be efficiently implemented on a micro-controller. We define a micro-controller model to compare the cost of DRECON-AES with the first-order masking of AES. We use an 8-bit micro-controller model [11] which takes:

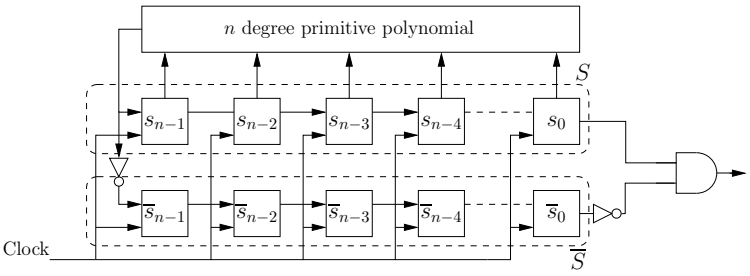


Fig. 4. Tweak Generation for DRECON

- load/store from/to RAM takes one clock cycle,
- load/store from/to ROM takes two clock cycles,
- XOR two registers takes one clock cycle,
- shift by one bit takes one clock cycle.
- swapping two nibbles of a byte takes one clock cycle.

We assume that the tweaked-sbox is stored in ROM. The masked implementations are assumed to use *GLUT* (global look-up table) with the same input and the output mask (as it is first order protection). The GLUT is also stored in ROM. Key Expansion is considered precomputed and thus omitted.

8 × 8 DRECON-AES: In the given scenario, the cost of each AES sub-operation for a standard 8 × 8 AES in terms of number of cycles are [11]:

- *SubBytes* (SB): 80
- *ShiftRows* (SR): 24
- *MixColumns* (MC): 256
- *AddRoundKey* (ARK): 64
- Memory Size: 256 Bytes

In DRECON-AES, the linear operations are exactly same as AES. The *SubBytes* is the only component which has changed. For 8 × 8 DRECON-AES, the number of clock cycles required for the *SubBytes* operation is $SB^* = 96$. It also needs one SR and two ARK operations less than the standard AES. Thus, the full 8 × 8 DRECON-AES (without Key Expansion) would need $Encryption = 10 \times SB^* + 9 \times SR + 9 \times MC + 9 \times ARK$ i.e. 4056 clock cycles and 256 × 256 bytes or 64Kbytes of ROM.

For masking using GLUT approach, the total number of 8 × 8 sboxes are 256 which is same as 8 × 8 DRECON-AES. The number of cycles to compute *SubBytes* is also same as 8 × 8 DRECON-AES i.e. $SB^* = 96$ clock cycles. Apart from *SubBytes*, there is one initial masking and a final demasking. This is called Extra Mask Addition (MA) and takes as many clock cycles as ARK. At each round, the *MixColumns* and *ShiftRows* operations need to be performed for the mask also. Thus a total masked AES (without Key Expansion), would need $Encryption = 10 \times SB^* + 20 \times SR + 18 \times MC + 11 \times ARK + 2 \times MA$ i.e. 6880 clock cycles and 256 × 256 bytes or 64Kbytes of ROM. Total cost estimation of 8 × 8 DRECON-AES and masked AES is shown in Table 1.

Table 1. Cost Comparison for 8-bit Micro-controller in terms of number of clock cycles taken for various operations

Architecture	SB	SR	MC	ARK	ROM (bytes)	MA	Encryption
AES	80	24	256	64	256	0	4048
8 × 8 DRECON-AES	96	24	256	64	64K	0	4056
Masked AES	96	24	256	64	64K	64	6880

Table 2. Cost Comparison for 8-bit Micro-controller in terms of number of clock cycles taken for various operations

Architecture	SB	SR	MC	ARK	ROM (bytes)	MA	Encryption
4×4 DRECON-AES	256	24	256	64	128	0	5656
Masked Alternative	224	24	256	64	128	64	8160

4×4 DRECON-AES: Implementation of 4×4 DRECON-AES in 8-bit micro-controller is a bit more tricky. The tweaked-sbox comprising of 16 non-linear functions can be stored in 128-bytes of ROM. The tweak determines the correct sbox for the current operation by additional XOR (Figure 5). Since the ROM in micro-controllers are often organised in bytes, a conditional swap operation followed by AND operation with 0x0F determines the correct output nibble.

With the defined model in the beginning of Section 4.4 into consideration, we have derived the number of clock cycles for each encryption and compare with its masking counterpart. Table 2 gives the number of cycles for *SubBytes* (SB), *ShiftRows* (SR), *MixColumns* (MC), *AddRoundKeys* (ARK). The table also lists the ROM required for the implementations.

The 8×8 DRECON-AES has almost null performance overhead, while its masking counterpart has a significant performance overhead, both having a 256x memory overhead. The memory overhead of the 8×8 DRECON-AES can be further reduced to as low as 16x for 4×4 DRECON-AES only at the cost of slightly higher performance overhead. For both the cases, DRECON-AES has an advantage over its masking counterpart in terms of the performance while having same memory overhead.

4.5 Hardware Implementation DRECON-AES

In hardware, two implementation styles are followed for DRECON-AES. The first one is *parallel implementation* which is preferred when sbox is small as in

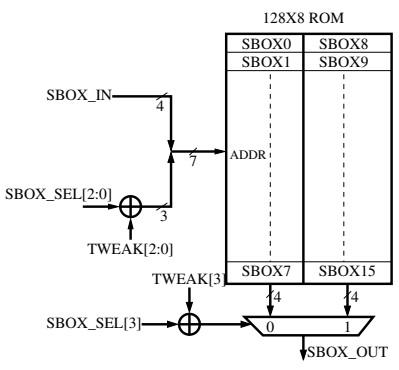


Fig. 5. 4×4 tweaked-sbox in Software

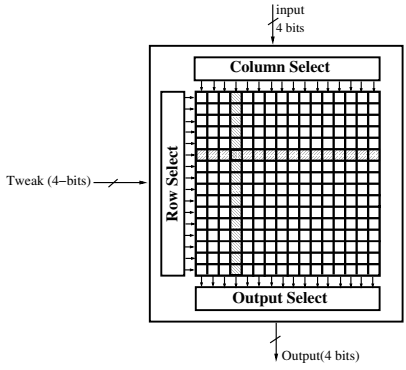


Fig. 6. 4×4 tweaked-sbox with in Hardware

Table 3. Comparing Resource Requirements for 8×8 DRECON-AES with Masking on an FPGA (XC5VLX50-2FF324)

Implementation	Slices	LUTs	Registers	Clock Cycles	Clock Period (ns)
4×4 AES[1]	1120	3472	1270	11	11.14
Masked 4×4 AES	3427	10589	1765	11	23.83
4×4 DRECON-AES	1379	3868	1583	11	10.3

[1] 4×4 AES is an implementation of the AES-128 algorithm with the 8×8 bit sbox replaced by a pair of 4×4 cryptographically strong sboxes.

the case of 4×4 DRECON-AES. The second is the *serialized implementation* used when sboxes are larger or when small area implementations are required. We adopted serialized implementation for 8×8 DRECON-AES.

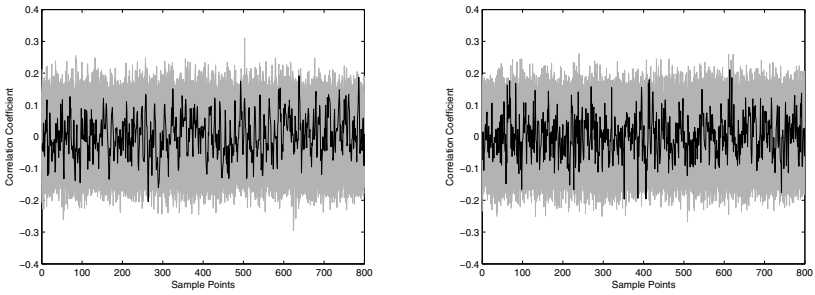
4×4 DRECON-AES: Much like the RSM countermeasure proposed in [26], DRECON makes excessive use of tables. This especially suits FPGA platforms which possess large memory blocks (BRAM) to implement arrays of sboxes. The BRAM are used to store the pool of sboxes ($\mathcal{F}$) efficiently. BRAM based unprotected implementation of ciphers have been shown to offer higher resistance against DPA as compared to other unprotected implementations [35]. The memory is addressed by a 4 bit tweak as shown in Figure 6. For DRECON-AES, we replicated this memory 32 times; once for each tweaked-sbox in the round. The value of the tweak is used to select a row while the input data selects a column in order to obtain the result. There are 32 such structures, one for each of the 32 substitution functions, present in the design. We use distributed RAM instead of BRAM to accelerate the attack. Resource requirements for the DRECON-AES implementation is compared with masked implementations of an equivalent AES with 4×4 sboxes in Table 3. The estimation for the masked implementation is computed from [32].

8×8 DRECON-AES: To implement 8×8 DRECON-AES, we followed the design of [23]. Serialised architecture with a rotating shift register and a single sbox is used. Each sbox access is split into two cycles. In the first cycle, two bytes - one from the state register and other from the tweak register - are applied to the input of the tweaked-sbox and its output is saved into the state byte. In the next cycle, both the state register and the tweak register are rotated by one byte. Thus, the *SubBytes* operation requires $2 \times 16 = 32$ clock cycles. The purpose of performing the state-byte update and rotation of state register in two different clock cycles is to be able to perform a sound security analysis of the implementation [23]. After the *SubBytes* operation, *ShiftRows*, *MixColumns* and *AddRoundKey* operations are performed in one clock cycle making a single round consisting of 33 clock cycles. The 8×8 tweaked-sbox consists of 2^8 8×8 sboxes. Each of the 2^8 sboxes was generated from AES sbox using the second strategy of Section 4.3. The resource requirement of 8×8 DRECON-AES implemented

Table 4. Comparing Resource Requirements for 8×8 DRECON-AES with Masking on an FPGA (XC5VLX50-2FF324)

Implementation	Slices	LUTs	Registers	Clock Period (ns)
Masked AES[1]	3948	13278	1592	14.955
8×8 DRECON-AES	1355	3716	1568	10.789

[1]The implementation exclude the PRNG used to generate mask.



(a) 1O correlation-collision attack (b) 2O correlation-collision attack

Fig. 7. Results of correlation-collision attack on 8×8 DRECON-AES using about $10,00,000$ traces. The correlation coefficients for wrong key differences are shown in grey and for the correct key difference (in this case $k_1 \oplus k_2 = 108$) in black.

in SASEBO-GII FPGA platform is shown in Table 4. The table also shows the resource requirement of the second scheme of [15] implemented in the same platform.

4.6 Attack on the Hardware Implementation

Recently correlation-collision attacks [24,22,23] have become a very effective tool to expose the vulnerabilities of 1O masking scheme in the presence of glitches. In [23], the authors have discussed the security of several ROM-based masking scheme of AES against correlation-collision attacks. In this section, we provide a similar security analysis of 8×8 DRECON-AES.

In *correlation-collision* attack [24], two sets of leakages of the same sbox instant during two different clock cycles with two different inputs, $x_i \oplus k_i$ and $x_j \oplus k_j$, are compared using a statistical test. If the two sets of leakages are similar in some statistical sense, a collision between the two sets are detected by the statistical test. The collision assures the relation $x_i \oplus k_i = x_j \oplus k_j$ or $k_i \oplus k_j = x_i \oplus x_j$. Correlation-collision attack was originally proposed to detect collision using the correlation between the mean values of the two sets of leakages. However in [22,23], it has been used to consider the higher order moments of the leakages also.

To verify the resistance of 8×8 DRECON-AES against DPA attack in the presence of glitches, we performed correlation-collision attack using both $1O$ and the $2O$ moments as described in [22]. For this evaluation also, SASEBO-GII board was used. The algorithm was implemented in Virtex 5 XC5VLX50 FPGA of SASEBO-GII which is driven by a clock frequency of 2 MHz. The power traces were acquired using Tektronix MSO 4034B Oscilloscope at the rate of 2.5 GS/s i.e. 1, 250 samples per clock period. Figure 7 shows the result of both $1O$ and the $2O$ correlation-collision attack on the leakages of the first and the second sbox access of the second round 8×8 DRECON-AES using about $10,00,000$ traces. It may be noted that similar implementations of several masking schemes have been reported to be vulnerable to correlation-collision attack in [23].

5 Conclusion

DRECON provides a simple and efficient method of constructing block ciphers with inherent and provable security against DPA. The use of off-the-shelf sboxes ensures that the cipher is secure against classical cryptanalysis. In a glitch-free scenario, the security against DPA is proved to be equal to first-order Boolean masking from an information theoretic perspective. Its resistance against univariate DPA is also proved in the presence of glitches. Additionally, the first-order and second-order univariate DPA security is validated empirically with implementations on the SASEBO GII side-channel evaluation board. From the implementation perspective, DRECON has several advantages over standard countermeasures such as masking, hiding, and re-keying. In future, we hope to extend DRECON to provide security against higher-order DPA attacks as well.

References

1. Abdalla, M., Bellare, M.: Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 546–559. Springer, Heidelberg (2000)
2. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener (ed.) [40], pp. 398–412
3. Clavier, C., Coron, J.S., Dabbous, N.: Differential Power Analysis in the Presence of Hardware Countermeasures. In: Koç, Ç.K., Paar (eds.) [12], pp. 252–263
4. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Improved Collision-Correlation Power Analysis on First Order Protected AES. In: Preneel, B., Takagi, T. (eds.) [30], pp. 49–62
5. Cover, T.M., Thomas, J.A.: Elements of Information Theory, 2nd edn. Series in Telecommunications and Signal Processing. Wiley-Interscience (July 2006)
6. Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.-X.: Block Ciphers That Are Easier to Mask: How Far Can We Go? In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 383–399. Springer, Heidelberg (2013)
7. Goubin, L., Patarin, J.: DES and Differential Power Analysis (The "Duplication" Method). In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)

8. Guajardo, J., Mennink, B.: On side-channel resistant block cipher usage. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 254–268. Springer, Heidelberg (2011)
9. Guilley, S., Sauvage, L., Flament, F., Vong, V.N., Hoogvorst, P., Pacalet, R.: Evaluation of Power Constant Dual-Rail Logics Countermeasures against DPA with Design Time Security Metrics. IEEE Trans. Computers 59(9), 1250–1263 (2010)
10. Herbst, C., Oswald, E., Mangard, S.: An AES Smart Card Implementation Resistant to Power Analysis Attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252. Springer, Heidelberg (2006)
11. Hoheisel, A.: Side-Channel Analysis Resistant Implementation of AES on Automotive Processors. Master's thesis, Ruhr-University Bochum, Germany (June 2009)
12. Paar, C., Koç, Ç.K. (eds.): CHES 2000. LNCS, vol. 1965. Springer, Heidelberg (2000)
13. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener (ed.) [40], pp. 388–397
14. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002)
15. Maghrebi, H., Prouff, E., Guilley, S., Danger, J.-L.: A first-order leak-free masking countermeasure. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 156–170. Springer, Heidelberg (2012)
16. Mangard, S., Popp, T., Gammel, B.M.: Side-Channel Leakage of Masked CMOS Gates. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 351–365. Springer, Heidelberg (2005)
17. Mangard, S., Pramstaller, N., Oswald, E.: Successfully Attacking Masked AES Hardware Implementations. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 157–171. Springer, Heidelberg (2005)
18. McEvoy, R.P., Tunstall, M., Whelan, C., Murphy, C.C., Marnane, W.P.: All-or-Nothing Transforms as a Countermeasure to Differential Side-Channel Analysis. IACR Cryptology ePrint Archive 2009, 185 (2009)
19. Medwed, M., Standaert, F.X., Großschädl, J., Regazzoni, F.: Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 279–296. Springer, Heidelberg (2010)
20. Messerges, T.S.: Using Second-Order Power Analysis to Attack DPA Resistant Software. In: Koç, Ç.K., Paar (eds.) [12], pp. 238–251
21. Micali, S., Reyzin, L.: Physically Observable Cryptography (Extended Abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
22. Moradi, A.: Statistical Tools Flavor Side-Channel Collision Attacks. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 428–445. Springer, Heidelberg (2012)
23. Moradi, A., Mischke, O.: How Far Should Theory Be from Practice? - Evaluation of a Countermeasure. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 92–106. Springer, Heidelberg (2012)
24. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-Enhanced Power Analysis Collision Attack. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010)
25. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011)

26. Nassar, M., Souissi, Y., Guilley, S., Danger, J.L.: RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. In: Rosenstiel, W., Thiele, L. (eds.) DATE, pp. 1173–1178. IEEE (2012)
27. Nikova, S., Rijmen, V., Schläffer, M.: Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. J. Cryptology 24(2), 292–321 (2011)
28. Kocher, P.C.: Leak-Resistant Cryptograhic Indexed Key Update, US Patent 6539092 (2003)
29. Piret, G., Roche, T., Carlet, C.: PICARO – A Block Cipher Allowing Efficient Higher-Order Side-Channel Resistance. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 311–328. Springer, Heidelberg (2012)
30. Preneel, B., Takagi, T. (eds.): CHES 2011. LNCS, vol. 6917, pp. 2011–2013. Springer, Heidelberg (2011)
31. Prouff, E., Roche, T.: Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols. In: Preneel, B., Takagi, T. (eds.) [30], pp. 63–78
32. Regazzoni, F., Yi, W., Standaert, F.X.: FPGA Implementations of the AES Masked Against Power Analysis Attacks. In: Proceedings of 2nd International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE) (February 2011)
33. Research Center for Information Security National Institute of Advanced Industrial Science and Technology: Side-channel Attack Standard Evaluation Board SASEBO-GII Specification, Version 1.01 (2009)
34. Rivain, M., Prouff, E., Doget, J.: Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 171–188. Springer, Heidelberg (2009)
35. Shah, S., Velegalati, R., Kaps, J.P., Hwang, D.: Investigation of DPA Resistance of Block RAMs in Cryptographic Implementations on FPGAs. In: Prasanna, V.K., Becker, J., Cumplido, R. (eds.) ReConFig, pp. 274–279. IEEE Computer Society (2010)
36. Standaert, F.X., Pereira, O., Yu, Y., Quisquater, J.J., Yung, M., Oswald, E.: Leakage Resilient Cryptography in Practice. Cryptology ePrint Archive, Report 2009/341 (2009), http://eprint.iacr.org/
37. Tiri, K., Akmal, M., Verbauwhede, I.: A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In: ESSCIRC 2002, pp. 403–406 (2002)
38. Tiri, K., Verbauwhede, I.: A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In: DATE, pp. 246–251. IEEE Computer Society (2004)
39. Waddle, J., Wagner, D.: Towards Efficient Second-Order Power Analysis. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 1–15. Springer, Heidelberg (2004)
40. Wiener, M. (ed.): CRYPTO 1999. LNCS, vol. 1666. Springer, Heidelberg (1999)

Appendix A: Proof of Lemma 1

Without loss of generality, we assume that the sbox in the statement of Lemma 1 belongs to the second round. Thus, the random variable Z is the output of a MDS mapping M of the diffusion layer in the first round (see Figure 1) which takes the outputs of m sboxes of the first round as inputs. Let $\boldsymbol{S}(X_{i_1}, T_{i_1})$, $\boldsymbol{S}(X_{i_2}, T_{i_2})$, $\ldots$, $\boldsymbol{S}(X_{i_m}, T_{i_m})$ be the inputs to the the MDS mapping. Since the operation is a MDS mapping, it can be realised by a $q \times q$ matrix whose elements are essentially non-zero. Thus Z can be represented as $a_1 \cdot \boldsymbol{S}(X_{i_1}, T_{i_1}) \oplus \cdots \oplus a_m \cdot \boldsymbol{S}(X_{i_m}, T_{i_m})$ where $a_j \in \{0,1\}^n \setminus \{0\}^n$. We can now compute the posterior probability of $Z = z$ given $X_1, \ldots, X_q, T$ as

$$\begin{aligned}\mathsf{P}[Z = z | X_1 = x_1, \ldots, X_q = x_q, T = t] =& \mathsf{P}[Z = z | X_{i_1} = x_{i_1}, \ldots, X_{i_m} = x_{i_m}, T = t] \\ =& \mathsf{P}[a_1 \cdot \boldsymbol{S}(X_{i_1}, T_{i_1}) \oplus \cdots \oplus a_m \cdot \boldsymbol{S}(X_{i_m}, T_{i_m}) = z \\ & | X_{i_1} = x_{i_1}, \ldots, X_{i_m} = x_{i_m}, T = t] \\ =& \mathsf{P}[a_1 \cdot \boldsymbol{S}(x_{i_1}, T_{i_1}) \oplus \cdots \oplus a_m \cdot \boldsymbol{S}(x_{i_m}, T_{i_m}) = z \\ & | T = t]\end{aligned}$$

The variable T may or may not belong to $\{T_{i_1}, \cdots, T_{i_m}\}$. Let us first assume that T does not belong to $\{T_{i_1}, \cdots, T_{i_m}\}$. In that case, the above probability can be given by

$$\mathsf{P}[Z = z | X_1 = x_1, \ldots, X_q = x_q, T = t] = \mathsf{P}[a_1 \cdot \boldsymbol{S}(x_{i_1}, T_{i_1}) \oplus \cdots \oplus a_m \cdot \boldsymbol{S}(x_{i_m}, T_{i_m}) = z]$$

Since, the sbox $\boldsymbol{S}(\cdot, \cdot)$ satisfies Proposition 2 and T_is are uniformly random, the variable $a_1 \cdot \boldsymbol{S}(x_{i_1}, T_{i_1}) \oplus \cdots \oplus a_m \cdot \boldsymbol{S}(x_{i_m}, T_{i_m})$ is also uniformly random and consequently $\mathsf{P}[Z = z | X_1 = x_1, \ldots, X_q = x_q, T = t] = 1/2^n$.

On the other hand, if T belongs to $\{T_{i_1}, \cdots, T_{i_q}\}$, let say $T = T_{i_1}$, the posterior probability of $Z = z$ can be given by

$$\begin{aligned}\mathsf{P}[Z = z | X_1 = x_1, \ldots, X_q = x_q, T = t] =& \mathsf{P}[a_1 \cdot \boldsymbol{S}(x_{i_1}, T_{i_1}) \oplus \cdots \oplus a_m \cdot \boldsymbol{S}(x_{i_m}, T_{i_m}) = z \\ & | T_{i_1} = t] \\ =& \mathsf{P}[a_1 \cdot \boldsymbol{S}(x_{i_1}, t) \oplus \cdots \oplus a_m \cdot \boldsymbol{S}(x_{i_m}, T_{i_m}) = z] \\ =& \mathsf{P}[a_2 \cdot \boldsymbol{S}(x_{i_2}, T_{i_2}) \oplus \cdots \oplus a_m \cdot \boldsymbol{S}(x_{i_m}, T_{i_m}) = \\ & z \oplus a_1 \cdot \boldsymbol{S}(x_{i_1}, t)]\end{aligned}$$

Since $m \geq 2$, this probability is also equates to $1/2^n$. Thus, in both the cases the posterior probabilities of $Z = z$ is $1/2^n$ which is equals to its a priori probability $\mathsf{P}[Z = z]$. Thus we conclude that Z is independent of the joint distribution of the variables $X_1, \ldots, X_q$ and T.

Appendix B: Algorithm of 4×4 DRECON-AES

Algorithm 1: Compute 4×4 DRECON-AES

Input: T: Master Tweak, ntr: number of encryption, P: Array of ntr plaintext
Output: C: Array of ntr ciphertext

begin
  Generate Key Expansion Tweak $T_K \in [0, 63]$ from Master Tweak T
  Generate 16 Tweaked Sboxes
  KeyExpansion $k[1] \dots k[9]$

  for $i = 1$ *to* ntr **do**
    Generate: $128 - bit$ Session Tweak T_S
    $state \leftarrow P[i]$
    for *round* $r = 1$ *to* 9 **do**
      for *nibble* $n = 1$ *to* 32 **do**
        $Sbox \leftarrow Sboxes[T_S[(4 * n : 4 * n + 3) mod 16]]$
        $state[n] \leftarrow Sbox(state[n])$ (SubBytes)
      end
      $state \leftarrow ShiftRows(state)$
      $state \leftarrow MixColumns(state)$
      $state \leftarrow AddRoundKey(state, k[r])$
    end
    for *nibble* $n = 1$ *to* 32 **do**
      $Sbox \leftarrow Sboxes[T_S[(4 * n : 4 * n + 3) mod 16]]$
      $state[n] \leftarrow Sbox(state[n])$ (Final SubBytes)
    end
    $C[i] \leftarrow state$
  end
  return C
end