On the Vulnerability of Low Entropy Masking Schemes

Xin Ye and Thomas Eisenbarth

Worcester Polytechnic Institute, Worcester, MA, USA
{xye,teisenbarth}@wpi.edu

Abstract. Low Entropy Masking Schemes (LEMS) have been proposed to offer a reasonable tradeoff between the good protection against side-channel attacks offered by masking countermeasures and the high overhead that results from their implementation. Besides the limited analysis done in the original proposals of LEMS, their specific leakage characteristics have not yet been analyzed. This work explores the leakage behavior of these countermeasures and shows two different methods how the leakage can be exploited, even by generic univariate attacks. In particular, an attack that exploits specific properties of RSM for AES as well as a more generic attack making very little assumptions about the underlying LEMS are introduced. All attacks are practically verified by applying them to publicly available leakage samples of the RSM countermeasure.

1 Motivation

Side channel attacks such as Power and EM analysis are still a major concern for embedded cryptographic solutions, in particular for cryptographic smart cards. One of the earliest and most studied countermeasure techniques is *masking* [4, 7, 20]. Masking, if done correctly, significantly increases the complexity of a successful attack. Results go as far as proving the impossibility of first-order attacks for appropriately masked implementations and leakages covered in the corresponding assumptions. This means that the simplest and probably most popular attacks such as classical DPA [8], CPA [3] and MIA [6] become impractical or in many cases even infeasible.

One major drawback of masking schemes is the significant overhead needed for their realization, especially for popular ciphers such as the AES. Computational overheads are usually significant and are due to mask processing and other adjusted computation such as just-in-time recomputation of look-up tables, or their redundant storage. In addition, masking schemes usually assume uniformly distributed random masks, i.e. a sufficiently good randomness generator has to be implemented and queried in addition to the protected cryptographic scheme. Hence, masking usually adds significant time and space overhead to cryptographic implementations in both hardware and software. Motivated by this overhead is the idea of reducing the entropy of the mask. Using fewer mask values can reduce the number of special cases that have to be handled by the implementation, allowing for a possible tradeoff between side channel resistance

and performance. While one could argue that this is a common approach in protecting logic styles [15], only limited work has been proposed to apply low-entropy masking at the architecture level [12, 13, 2]. These works claim that first-order attacks are still prevented. However, while analysis of masking logic styles suggests remaining leakage [17], no deeper analysis of possible weaknesses has been performed. This work takes a first systematic step in that direction.

Contribution In this paper we formalize low entropy masking schemes and reveal the vulnerability of their limited protection. We propose two leakage composition based attacks to show how to exploit the weakened leakage even by univariate adversaries. Experiments are performed on a software implementation of Rotating Sboxes Masking (RSM) to verify the validity of the proposed attacks.

The rest of the paper is organized as follows: Section 2 reviews the low entropy masking schemes and gives the adversarial model for these schemes; Section 3 introduces the leakage distribution decomposition attack (LDDA); Section 4 proposes the leaking set collision attack (LSCA); experiments are carried out in Section 5 to verify LDDA and LSCA using measurements from DPA contest V4; and finally conclusions are made in Section 6.

2 Background

In the following we give a detailed definition of low-entropy masking schemes and the assumed adversarial model and show why low-entropy masking schemes can thwart standard attacks like DPA and CPA.

2.1 Low Entropy Masking Schemes

Low entropy masking schemes (LEMS) are a countermeasure against side channel attacks (SCA). Like other masking schemes, they try to randomize the observed leakage by applying random values to intermediate states. However, the size of the mask alphabet is reduced, resulting in a limited extend of randomization of leaking states. For example, for a LEMS protected implementation of AES, the mask set $\mathcal{M}$ is a strict subset of $\{0,1\}^8$ such that the number of applicable mask values is much smaller than 256. The Rotating SBoxes Masking (RSM) scheme proposed in [13] is a realization of LEMS. It is a Boolean masking scheme that uses 16 mask values uniformly at random to protect AES internal states. In general, we denote the set of masks $\mathcal{M} = \{m_1, m_2, ..., m_s\} \subset \mathbb{F}_2^n$. We say a LEMS has masking entropy of $\log s$ if mask values are chosen uniformly at random from this set. The RSM is therefore said to have 4 bits of mask entropy. Furthermore, authors in [12] proposed a selection criterion of optimal mask values for LEMS. According to this guideline the following 16 byte values (written in hex format) are used as the mask set in the DPA contest V4.

$$\mathcal{M} = \{00,0F,36,39,53,5C,65,6A,95,9A,A3,AC,C6,C9,F0,FF\}$$

The 16 chosen values form an $[8, 4, 4]$ linear code. It is therefore not surprising that they satisfies the *self-complementary* property: $\overline{\mathcal{M}} = \mathcal{M}$. Namely, $m \in \mathcal{M}$ if and only if $\bar{m} \in \mathcal{M}$, where $\bar{m}$ is the bitwise inversion of m.

The benefit of applying LEMS lies in the fact that it saves lots of computation when compared to a full entropy masking scheme (FEMS) where $s = 2^n$. The latter usually suffers from the huge amount of additional computation as a consequence of repeated masking/de-masking for the non-linear operation of a block cipher (e.g. Sbox in AES). One example is the Generalized Look-Up Table countermeasure proposed in [16]. It increases the size of a single Sbox sufficiently to make parallelized implementation of AES on FPGAs infeasible. However, with fewer masks, the total number of necessary extra-computations can be kept at an acceptable level or even completed from pre-computations (e.g. defining masked sbox as look up tables). In short, LEMS enables more efficient implementation of a masking countermeasure. Unavoidably, applying LEMS causes some loss of protection when compared to FEMS. The natural question is how much security has been sacrificed and whether an attacker can construct an efficient attack to break the LEMS. Experiments in [13] show that RSM can resist univariate attacks including first and second-order DPA and CPA. The work uses MIA as the metric to get a quantification of 0.015 bit of information leakage in the described experimental setup, motivating a claim that such a low amount should be hard to exploit.

2.2 Adversarial model

We assume an adversarial model with the following notations. Let X be the partial input/output (i.e. knowntext) of the algorithm known to the adversary (e.g. one byte of plaintext of encryption), k be the partial key in use, $Y = f_k(X)$ be the sensitive algorithmic state value to be protected. Here f is the target function which is usually a part of the algorithm. Let $\mathcal{M} = \{m_1, ..., m_s\}$ be the set of masks. When $M \in \mathcal{M}$ is applied in a first-order masking scheme $\text{MASK}(\cdot, \cdot)$, $Y_M = \text{MASK}(Y, M)$ is generated internally to protect the sensitive Y. The masked output Y_M is also called the leaking value (or masked state). The observed univariate leakage Λ is considered as the functional evaluation of the leaking value Y_M in the leakage function $L(\cdot)$ as expressed in equation (1).

$$\Lambda = L(Y_M) = L(\text{MASK}(f_k(X), M)) \tag{1}$$

The set of all leaking values for Y is denoted as $Y_{\mathcal{M}} = \{Y_{m_1}, ..., Y_{m_s}\}$. A sensitive internal state y is protected by leaking values y_{m_i} with equal probability because the mask is chosen uniformly at random from $\mathcal{M}$.

In sum, the knowledge of the adversary includes the input X, target function f, and the univariate leakage Λ for processing leaking values Y_M. Our first attack in Section 3 also assumes the adversary to know the mask set $\mathcal{M}$, while in our second attack in Section 4 we only assume the mask set to satisfy the self-complementary property; the attacker does not necessarily need to know the mask values.

3 Leakage Distribution Decomposition Attack

LEMS are designed to resist low statistical order DPA/CPA attacks while maintaining small computational overhead. The low level of leakage indicated by the mutual information $I(HW(Y_M); Y)$, as quantified in [13, 12], however, does not exclude the possibility of a univariate attack. In this section we analyze the composition of the leakage distribution under the protection of LEMS. We propose a univariate attack that can correctly decompose the observed one-dimensional distribution of leakage into several sub-distributions.

3.1 Leakage Distribution Composition

With the masking countermeasure, one algorithmic internal state Y can produce side channel leakage Λ through multiple leaking values $Y_{m_1}, ..., Y_{m_s}$. Consequently, the conditional entropy of leakage $H(\Lambda \mid Y)$ increases, making the classical attacks harder to succeed. According to equation (1) the leakage Λ depends on the knowntext X and the mask M, which are the main sources of entropy. If the knowntext is fixed to one value $X = x$ at a time, the leakage entropy is lowered because only mask values are changed and LEMS only contains a small number of masks.

We use $\mathcal{D}^{X=x}_{M\in\mathcal{M}}[\Lambda]$ (or simply $\mathcal{D}^{x}_{\mathcal{M}}[\Lambda]$) to denote the leakage distribution under the condition that the knowntext X is fixed to x and the mask M is chosen uniformly at random from the mask set $\mathcal{M}$. In this situation, $X = x$ implies only one sensitive value $y = f_k(x)$ is to be protected by the masks, which results in the leaking set $(y)_{\mathcal{M}}$. Processing each leaking value y_{m_i} produces leakage $L(y_{m_i})$. The respective leakage observations form a leakage sub-distribution denoted by $\mathcal{D}^{X=x}_{M=m_i}[\Lambda]$ (or simply $\mathcal{D}^{x}_{m_i}[\Lambda]$) [1]. Since the leaking set $(y)_{\mathcal{M}}$ contains s leaking values, the observed leakage distribution $\mathcal{D}^{x}_{\mathcal{M}}[\Lambda]$ is a composition of s sub-distributions, namely,

$$\mathcal{D}^{x}_{\mathcal{M}}[\Lambda] = \frac{1}{s}\sum_{i=1}^{s}\mathcal{D}^{x}_{m_i}[\Lambda] = \frac{1}{s}\sum_{i=1}^{s}\mathcal{D}[L(y_{m_i})] \tag{2}$$

This equality actually comes from the law of total probability, i.e.

$$\begin{aligned} p[\Lambda = \lambda \mid X = x] &= \sum_{i=1}^{s} p[\Lambda = \lambda \mid M = m_i, X = x] \cdot Pr[M = m_i] \\ &= \frac{1}{s}\sum_{i=1}^{s} p[\Lambda = \lambda \mid M = m_i] \end{aligned}$$

simply because $\mathcal{D}^{x}_{\mathcal{M}}[\Lambda]$ has the same meaning as the pmf/pdf $p[\Lambda = \lambda \mid X = x]$ and $\mathcal{D}^{x}_{m_i}[\Lambda]$ the same as $p[\Lambda = \lambda \mid M = m_i, X = x]$.

[1] The notation $\mathcal{D}^{X=x}_{M=m_i}[\Lambda]$ is of the same meaning of leakage distribution as $\mathcal{D}[L(y_{m_i})]$. Both describe the leakage for processing y_{m_i}. The former emphasizes leakage decomposition and the latter focuses on connecting with estimated sub-distributions.

It is important to see that in LEMS the distribution $\mathcal{D}^x_{\mathcal{M}}[\Lambda]$ with fixed input x is different from the overall leakage distribution $\mathcal{D}[\Lambda]$ where the knowntext is not fixed. The former is a mixture of only s sub-distributions, while the latter is composed of all 2^n sub-distributions caused by all 2^n leaking values. In fact, the proposed leakage distribution decomposition attack (LDDA) makes use of this difference to explore the weakness of LEMS. It also indicates that the univariate LDDA cannot be extended to attack FEMS where both $\mathcal{D}^x_{\mathcal{M}}[\Lambda]$ and $\mathcal{D}[\Lambda]$ are composed of 2^n sub-distributions and hence not distinguishable from each other.

3.2 Procedure of Leakage Distribution Decomposition Attack

Prior to the attack, the adversary needs to estimate the sub-distributions $\mathcal{D}[L(v)]$ of leakage for each leaking value v. We discuss this issue in more detail in Section 3.3 and 3.4. Here the attacker is assumed to have already obtained a precise estimation of sub-distributions. We show how this idea of decomposition in leakage distribution converts to a side channel attack. For each subkey hypothesis g and each prefixed knowntext $X = x$, the adversary follows a three-step procedure.

1. Find the hypothetical leaking set $(\hat{y})_{\mathcal{M}}$;
2. Compute the hypothetical mixture $\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{\Lambda}]$;
3. Evaluate the distance $\texttt{Dist}(\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{\Lambda}] \| \mathcal{D}^x_{\mathcal{M}}[\Lambda])$ between the mixture and the observed distribution.

More specifically, with the subkey hypothesis g for a subkey k, the adversary computes $\hat{y} = f_g(x)$ and its respective masked states $\hat{y}_{m_i} = \text{MASK}(\hat{y}, m_i)$ for all $m_i \in \mathcal{M}$. Since each hypothetical leaking value y_{m_i} contributes as one component $\hat{\mathcal{D}}[L(\hat{y}_{m_i})]$ of the leakage distribution, the adversary rebuilds the hypothetical mixture of all the s sub-distributions as

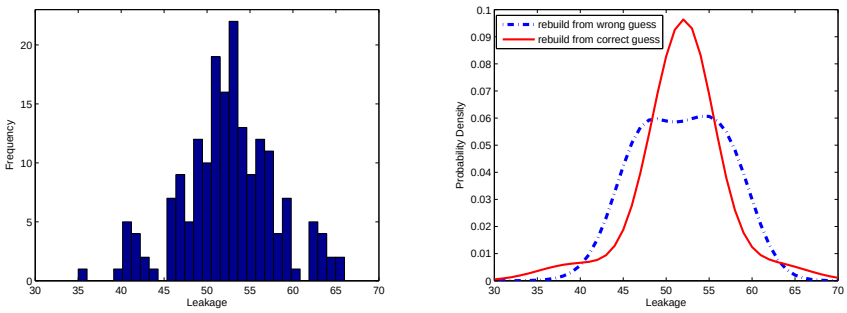
$$\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{\Lambda}] = \frac{1}{s} \sum_{i=1}^{s} \hat{\mathcal{D}}[L(\hat{y}_{m_i})] \tag{3}$$

Next, the adversary measures the similarity of the hypothetical mixture $\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{\Lambda}]$ and the observed distribution $\mathcal{D}^x_{\mathcal{M}}[\Lambda]$. A distance metric $\texttt{Dist}(\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{\Lambda}] \| \mathcal{D}^x_{\mathcal{M}}[\Lambda])$ is evaluated for this purpose. In general, a small value of the computed distance metric indicates the two distributions are close to each other. A typical instantiation of the distance metric is the Kolmogorov-Smirnov distance suggested by [22, 23], which is later used in our experiments.

The adversary repeats the three-step procedure for all subkey hypotheses and all prefixed x. Her final decision for the correct subkey k is the hypothesis k^* that results in the lowest averaged distance as in equation (4). The attack is successful if $k^* = k$.

$$k^* = \underset{g}{\text{argmin}} \left\{ \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \texttt{Dist}(\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{\Lambda}] \| \mathcal{D}^x_{\mathcal{M}}[\Lambda]) \right\} \tag{4}$$

Please note that the LDDA does not predict each individual *leaking state*. Instead, it analyzes the entire predicted *leaking set*. Figures 1(a) and 1(b) give an intuitive idea of how the decomposition of the observed distribution works for correct and incorrect subkey guesses.



(a) (b)

Fig. 1. The observed leakage distribution (a) VS the hypothetical mixtures (b). The rebuilt mixture from the correct guess has a similar shape to the observed distribution while the mixture from the wrong guess is quite different from the observed distribution.

Validity If the subkey guess g is correct, i.e. $g = k$, then $\hat{y} = f_g(x) = f_k(x) = y$ and the prediction of leaking set is correct $(\hat{y})_{\mathcal{M}} = (y)_{\mathcal{M}}$. Given precise estimations of sub-distributions, the rebuilt mixture $\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{\Lambda}]$ from equation (3) will be close to the observed distribution $\mathcal{D}^x_{\mathcal{M}}[\Lambda]$ because

$$\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{\Lambda}] = \frac{1}{s}\sum_{i=1}^{s}\hat{\mathcal{D}}[L(\hat{y}_{m_i})] = \frac{1}{s}\sum_{i=1}^{s}\mathcal{D}[L(y_{m_i})] = \mathcal{D}^x_{\mathcal{M}}[\Lambda]$$

However if the hypothesis is wrong, i.e. $g \neq k$, then $\hat{y} = f_g(x) \neq f_k(x) = y$ for most of the inputs x, hence the hypothetical leaking set $(\hat{y})_{\mathcal{M}}$ has a low probability[2] to be the same as the actual leaking set $(y)_{\mathcal{M}}$. It follows that with high probability the rebuilt mixture $\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{\Lambda}]$ differs significantly from the observed $\mathcal{D}^x_{\mathcal{M}}[\Lambda]$ and hence their distance metric output should be large.

3.3 LDDA With Profiling

We have mentioned that the adversary should estimate the sub-distributions before mounting the LDDA attack. This has a straightforward solution by combining a profiling phase. More specifically, the profiling adversary is also assumed

[2] An exception is when $(\hat{y})_{\mathcal{M}}$ is a permutation of $(y)_{\mathcal{M}}$ for some particular g and x. Such exception occurs with small probability because the predicted leaking states take the range of entire $\{0,1\}^n$ rather than $\mathcal{M}$

to have full control of the masks during the profiling stage – she knows each mask that is applied in each invocation. This assumption is frequently used in previous work [14, 9, 18]. The described attacks require at least bi-variate leakages consisting of the sample for processing the mask and the sample for the masked state. Hence, these approaches are not applicable for univariate attacks. Nevertheless, the profiling capability allows the adversary to build univariate leakage templates for each leaking value $v = \text{MASK}(f_{k'}(x), m)$ on another device that runs the same crypto algorithm with a different but known key k'. In other words, although the low entropy masking protection mingles different sub-distributions $\mathcal{D}[L(v)]$ together to achieve confusion, the assumed profiling adversary can still isolate each from the mixture. The isolated $\mathcal{D}[L(v)]$ can then serve as a sub-distribution look up table, enabling the adversary to rebuild the hypothetical mixture (the second step of LDDA) easily.

3.4 LDDA Without Profiling

Allowing the adversary having profiling capability is sometimes demanding. We show that sub-distribution estimation is also feasible for adversaries who are not granted with such privilege. This is achieved by assuming a leakage model and estimating the expression of leakage function explicitly. For a clear illustration, we assume the commonly accepted Hamming weight leakage model for a LEMS protected AES. It should be mentioned that advanced techniques of non-profiling leakage modeling such as linear regression model [5] may play a similar role if adjusted properly. With the Hamming weight model, the leakage Λ is expressed as a linear function of the Hamming weight of the leaking states with additive white Gaussian noise ϵ. I.e.

$$\Lambda = aHW\left(Y_M\right) + b + \epsilon$$

where the coefficients a, b are unknown constants, and the noise $\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$ is mean zero and the noise level σ is also unknown. Since the sub-distributions $\mathcal{D}[L(v)]$ for processing leaking value v can now be represented as $\mathcal{N}(\Lambda; aHW(v)+ b, \sigma^2)$, estimating sub-distributions is simplified to estimating the unknown parameters a, b, σ. Meanwhile, it is easy to see that the overall leakage distribution $\mathcal{D}[\Lambda]$ is a weighted composition of nine Gaussian curves. I.e.

$$\mathcal{D}[\Lambda] \approx \sum_{h=0}^{8} w_h \mathcal{N}(\Lambda; ah + b, \sigma^2) \tag{5}$$

where $0 \leq w_h \leq 1$ is the proportion of the normal curve $\mathcal{N}(\Lambda; ah + b, \sigma^2)$ and $\sum_{h=0}^{8} w_h = 1$. It follows that the Hamming weight h of leaking values Y_M forms a Binomial distribution and the weight parameters $w_h = \binom{8}{h}/2^8$, provided that the knowntext X is uniformly distributed. It is because xoring and SBoxing are one-to-one mappings. They deliver the uniform distribution from X to the sensitive Y and its masked output Y_M.

Finally, we solve the parameter estimation as an optimization problem. Optimal choices of a, b, σ should minimize the difference between the two sides of equation (5), namely, the observed overall leakage distribution and the composition of the parameterized sub-distributions. We set it as the objective function in equation (6). Furthermore, the optimization should be associated with the restriction that the statistical characteristics of the two sides should be approximately equal as in (7). Examples of the restriction functions are the statistical moments including $\text{Mean}(\mathcal{D}[\Lambda]) \approx 4a + b$, $\text{Var}(\mathcal{D}[\Lambda]) \approx \sigma^2 + 2a^2$ (derived from analysis of variance) and etc. The optimally parameterized $\mathcal{N}(\Lambda; aHW(v)+b, \sigma^2)$ can then serve as a sub-distribution look up table, enabling the adversary to carry out the LDDA.

$$\texttt{Minimize Dist}(\mathcal{D}[\Lambda] \| \sum_{h=0}^{8} w_h \mathcal{N}(\Lambda; ah + b, \sigma^2)) \tag{6}$$

$$\texttt{StatChar}(\mathcal{D}[\Lambda]) \approx \texttt{StatChar}(\sum_{h=0}^{8} w_h \mathcal{N}(\Lambda; ah + b, \sigma^2)) \tag{7}$$

It should be mentioned that the non-profiling LDDA is heavily influenced by the accuracy of leakage modeling. Large bias results in the derived sub-distributions being significantly different from the actual leakage function and hence reduces the efficiency or even disables the LDDA.

4 Leaking Set Collision Attack

The previously discussed LDDA follows a 'decompose'-then-'rebuild' approach to compare the distributions of leakage. We now propose a second attack named leaking set collision attack (LSCA). It circumvents the 'rebuild' step and allows adversary directly comparing related distributions and therefore gains the benefit of avoiding the sub-distribution estimation.

4.1 Existence of Leaking Set Collisions

The approach extends side channel collision attacks [19, 11] by defining collisions between two leaking sets. Two distinct knowntexts $x \neq x'$ are said to induce a *leaking set collision* if the respective leaking sets are the same, i.e.

$$(y)_{\mathcal{M}} = \{y_{m_1}, ..., y_{m_s}\} = \{y'_{m_1}, ..., y'_{m_s}\} = (y')_{\mathcal{M}}$$

For Boolean masking schemes, the existence of leaking set collisions is a consequence of the *self-complementary property* for the choice of the mask values suggested in [12, 13]. It indicates that if m is chosen as a possible mask value, so should its bitwise inverse $\bar{m} = m \oplus 1^n$ as explained in Section 2.1 (1^n denotes the all-1 bit string, e.g. `0xff` for a byte). One simple choice is $y' = \bar{y}$. Because for any $m \in \mathcal{M}$,

$$\begin{aligned}(\bar{y})_m = \bar{y} \oplus m &= y \oplus 1^n \oplus m \\ &= y \oplus \bar{m} = y_{\bar{m}} \in (y)_{\mathcal{M}}\end{aligned}$$

This proves $(\bar{y})_{\mathcal{M}} \subset (y)_{\mathcal{M}}$. Similarly the other direction $(\bar{y})_{\mathcal{M}} \supset (y)_{\mathcal{M}}$ also holds and hence $(y)_{\mathcal{M}} = (\bar{y})_{\mathcal{M}}$. On the other hand, this choice $y' = \bar{y}$ identifies a relation between the respective knowntexts x, x' by setting $f_k(x') = \overline{f_k(x)}$. It is equivalent to

$$x' = f_k^{-1}(1^n \oplus f_k(x)) \tag{8}$$

It implies that the knowntext pair $\langle x, x' \rangle$ derived from equation (8) results in a leaking set collision between $(y)_{\mathcal{M}}$ and $(y')_{\mathcal{M}}$.

4.2 Building a Leaking Set Collision Attack

An importance consequence of the leaking set collision is that the respective underlying leakage distributions are identical. In fact, the set collision $(y)_{\mathcal{M}} = (y')_{\mathcal{M}}$ implies the both $\mathcal{D}^x_{\mathcal{M}}[\Lambda]$ and $\mathcal{D}^{x'}_{\mathcal{M}}[\Lambda]$ have the same composition of sub-distributions.

$$\mathcal{D}^x_{\mathcal{M}}[\Lambda] = \frac{1}{s}\sum_{i=1}^{s} \mathcal{D}[L(y_{m_i})] = \frac{1}{s}\sum_{i=1}^{s} \mathcal{D}[L(y'_{m_i})] = \mathcal{D}^{x'}_{\mathcal{M}}[\Lambda]$$

Therefore, the empirically observed leakage distributions $\mathcal{D}^x_{\mathcal{M}}[\Lambda]$ and $\mathcal{D}^{x'}_{\mathcal{M}}[\Lambda]$ should be very close to each other. We now show how to convert this into a side channel attack against LEMS protected AES. The Sbox of the first round is chosen as the target function. Hence, the sensitive states y, y' are the s-box outputs and the knowntexts x, x' are the corresponding plaintext byte values[3]. The paired relation in equation (8) is then instantiated as in the following pairing equality in (9).

$$x' = \texttt{Pairing}(x, k) = k \oplus S^{-1}(0\text{xff} \oplus S(x \oplus k)) \tag{9}$$

It indicates that the plaintext pair $\langle x, x' \rangle$ which satisfies the paring equality forms a leaking set collision at their respective masked outputs.

The adversary, however, does not know the subkey k and cannot directly plug in the pairing equality to derive a collision. Nevertheless, she can make subkey hypothesis g and check for collisions just like a standard side channel attacker. A detailed attacking procedures is shown in Algorithm 1. It firstly sorts all leakages according to their respective plaintext x so that the empirical distributions $\mathcal{D}^x_{\mathcal{M}}[\Lambda]$ are obtained for all possible x. The adversary then starts testing subkey hypotheses. With each hypothesis g, she computes the hypothetical pairing $x' = \texttt{Pairing}(x, g)$ defined in equation (9). The two sets of related leakage distributions $\mathcal{D}^x_{\mathcal{M}}[\Lambda]$ and $\mathcal{D}^{x'}_{\mathcal{M}}[\Lambda]$ are fetched and their similarity is measured using the distance metric `Dist()`. In practice, the adversary can add up the computed distances derived from all possible collisions (line 8 of the algorithm). The decision strategy is similar to LDDA: the adversary determines as

[3] The same approach can be applied to arbitrary intermediate states, as long as they are a non-linear function of x and k: For states y that are linear functions of x and k, e.g. the s-box input, the key cancels out so that the knowntext pair become independent from the key, making the conversion into an attack infeasible.

the correct subkey the hypothesis k^* that results in the smallest overall distance. The attack is successful if $k^* = k$.

Algorithm 1 Leaking Set Collision Attack on RSM-AES

Input: Number of traces q; Knowntexts $x_1, ..., x_q$; leakages $\lambda_1, ..., \lambda_q$
Output: Subkey Decision k^*

Precomputation:
for $x = 0$ **to** 255 **do**
  $\mathcal{D}^x_{\mathcal{M}}[\Lambda] = \{\lambda_i \mid x_i = x\}$ ▷ collect leakage whose knowntext is x
end for
Key recovery:
for $g = 0$ **to** 255 **do**
  $\delta_g = 0$
  for $x = 0$ **to** 255 **do**
    $x' = \texttt{Pairing}(x, g)$ ▷ compute hypothetical pairing x'
    $\delta_g = \delta_g + \texttt{Dist}(\mathcal{D}^x_{\mathcal{M}}[\Lambda] \| \mathcal{D}^{x'}_{\mathcal{M}}[\Lambda])$ ▷ sums the distances from all pairings
  end for
end for
$k^* = \text{argmin}_g\{\delta_g\}$
return k^*

Validity If the key hypothesis is correct, i.e. $g = k$, then the derived pairing $x' = \texttt{Pairing}(x, g) = \texttt{Pairing}(x, k)$ is exactly the same as the true pairing equality in equation (9). It follows that a leaking set collision $(y)_{\mathcal{M}} = (y')_{\mathcal{M}}$ is generated. Hence the compared distributions should feature a low distance metric quantity $\texttt{Dist}(\mathcal{D}^x_{\mathcal{M}}[\Lambda] \| \mathcal{D}^{x'}_{\mathcal{M}}[\Lambda])$. However if the subkey hypothesis is wrong, the computation yields

$$y' = S(x' \oplus k) = S(g \oplus S^{-1}(\text{0xff} \oplus S(x \oplus g)) \oplus k)$$

It is different from $\bar{y} = \text{0xff} \oplus S(x \oplus k)$ for most x. Hence the resulting leaking set $(y)_{\mathcal{M}}$ has low probability to completely overlap $(y')_{\mathcal{M}}$ and the two distributions have high probability to differ significantly.

Complexity It should be mentioned that the roles of x and x' of a hypothetical pairing are symmetric for any hypothesis. That is, if x' is a hypothetical pairing of x satisfying $x' = \texttt{Pairing}(x, g)$, then reversely x is also a pairing of x' satisfying $x = \texttt{Pairing}(x', g)$. Here is a simple proof.

$$\begin{aligned} x'' = \texttt{Pairing}(x', g) &= g \oplus S^{-1}(\text{0xff} \oplus S(x' \oplus g)) \\ &= g \oplus S^{-1}(\text{0xff} \oplus S(S^{-1}(\text{0xff} \oplus S(x \oplus g)))) \\ &= g \oplus (x \oplus g) = x \end{aligned}$$

This symmetry implies there are a total of 128 possible leaking set collisions for all 256 knowntexts x. It suffices to make only 128 distance comparisons for testing one hypothesis. Therefore the total complexity is 256×128 distance computations to recover one key byte.

Comparing LSCA with LDDA One common feature of LDDA and LSCA is that both attacks are achieved by comparing leakage distributions. More precisely, the compared leakage distributions refer to the leakages $\mathcal{D}^x_{\mathcal{M}}[\Lambda]$ with some prefixed knowntext x. It results in a lowered leakage entropy which become exploitable by the two attacks.

There are also many differences between the two attacks. Firstly, the LDDA compares empirically observed leakage distribution with the rebuilt hypothetical mixtures, while the LSCA compares two sets of distributions that are both obtained empirically. Therefore, the correct subkey hypothesis in the LDDA measures the closeness of the empirical distribution to its underlying distribution. In the LSCA it measures the closeness between two empirical distributions that are sampled from the same underlying distribution. Secondly, the LDDA requires sub-distribution estimations to complete the "rebuild" step, while the LSCA avoid this. We have seen that estimating sub-distributions not only adds some complexity or even requires profiling privilege, but is also influenced by the accuracy of leakage modeling. Thus the LSCA does not suffer from the modeling bias. Last but not the least, the LDDA requires the mask set $\mathcal{M}$ to be known but the LSCA only requires the self-complementary property for the masking set $\mathcal{M}$. To sum up, the LDDA shows the explicit composition of leakages and the LSCA makes use of leakage composition implicitly and is more efficient in practice.

5 Experiments

In this section, we carry out the LDDA and LSCA described in Section 3 and Section 4. Our experiments are performed on the measurements from DPA contest V4 [1]. It is a software implementation of AES-256 protected by the RSM countermeasure (cf. Section 2.1) and a total of 100,000 leakage measurements are provided. All attacks are performed on a univariate leakage sample representing the leakage of the first round AES output of SBox. Before showing our result we want to mention as reference that [13] shows 0 success rate for DPA,CPA and VPA based on 150,000 observations of a hardware implementation of RSM. It also reports 0.001 to 0.012 bit of information being leaked from mutual information analysis.

5.1 LDDA With Profiling

We firstly implemented LDDA using the template attack approach and we assume full knowledge of the mask application during the profiling stage as detailed in Section 3.3. A total of 50,000 measurements are used to build the templates,

i.e. the 256 sub-distributions $\mathcal{D}[L(v)]$ of leakages for processing each possible leaking state v. The obtained sub-distributions are represented as 256 Gaussian curves $\mathcal{N}(\Lambda; \mu_v, \sigma_v^2)$. Upon the completion of sub-distribution estimation, another 2,000 to 16,000 measurements are used to test all 256 subkey hypotheses using the 3-step LDDA. In particular, the rebuilt distribution from each hypothesis is now instantiated as a Gaussian mixture,

$$\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{\Lambda}] = \frac{1}{s} \sum_{v \in (\hat{y})_{\mathcal{M}}} \mathcal{N}(\hat{\Lambda}; \mu_v, \sigma_v^2)$$

resulting in a model similar to [10]. The Gaussian mixture is compared with the observed distribution $\mathcal{D}^x_{\mathcal{M}}[\Lambda]$ using Kolmogorov-Smirnov (KS) distance metric.

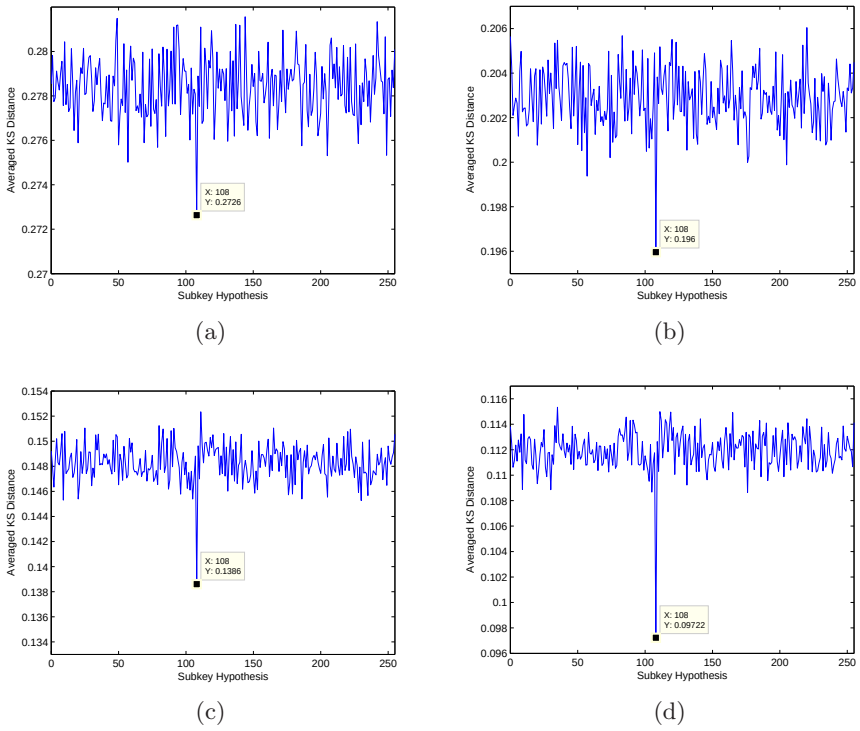


Fig. 2. Profiling LDDA hypothesis testing: Kolmogrov-Smirnov distance (y-axis) between observed leakage distribution and rebuilt Gaussian mixture from all subkey hypotheses (x-axis) with the profiled sub-distributions. Experiments use 2,000 traces (a); 4,000 traces (b); 8,000 traces (c); and 16,000 traces (d).

Figure 2 shows the profiling LDDA hypothesis testing for the first subkey byte. We can see that LDDA succeed – the correct subkey $k = 108$ always

gives the smallest KS distance among the 256 subkey hypotheses– whenever more than 2000 traces are used for testing. It verifies the correctness of the LDDA that only the correct hypothesis yields a correct decomposition of the leakage distribution. The four plots also show that the KS-distance drops when increasing the number of testing traces. In particular, the averaged KS distance for the correct subkey hypothesis drops from 0.273 all the way to 0.097. While for the wrong hypotheses, the average drops from 0.279 to around 0.112. The reason is that the computed distance depends on two main factors: *(1)* the correctness of the prediction of the leaking set $Y_{\mathcal{M}}$ (the effect exploited by LDDA); and *(2)* the sampling errors: viewing the observed leakage (the empirical one) as the samples from its underlying distribution (the true one approximated during the profiling). The law of large numbers implies that the empirical distribution of the leakage converges to its underlying distribution when increasing the number of leakages. Therefore, using more testing traces reduces the influence of the sampling error and hence decreases the overall KS distance metric. As a consequence, the correct hypothesis becomes better distinguishable from the wrong guesses.

5.2 LDDA Without Profiling

Our second group of experiments carries out the LDDA without a profiling stage as described in Section 3.4. Each experiments estimate different combinations of the required parameters in the presented optimization problem in equations (6) and (7). The guessing entropy from [21] is used for the evaluation of the attack. That is, the subkey k is said to have guessing entropy t if the KS distance for k is on average the t-th smallest value among all KS distances for all hypotheses. Results are summarized in Table 1.

Table 1. Performance evaluation using Guessing Entropy (GE) for the non-profiling LDDA. Best case is evaluated with optimal estimation of parameters; Worst case is with non-optimal estimation.

Number of Traces	20,000	40,000	60,000	80,000	100,000
GE (average case)	19.74	16.65	4.02	2.93	1.31
GE (worst case)	30	33	11	9	5
GE (best case)	9	2	2	1	1

It can be seen from the table that the correct subkey k has very low guessing entropy of 1 or 2 if more than 40,000 traces are used in the optimal estimation cases. Even for the worst estimation case shown in the table, the guessing entropy is still 33. The average estimation cases indicate that the non-profiling LDDA enables a reasonable attack – the guessing entropy is kept at an acceptable level– whenever more than 60,000 measurements are used.

It can be seen that the non-profiling LDDA needs much more traces to succeed comparing to the profiling LDDA. Notice that the latter serves as the closest

approximation to the real leakage function and the non-profiling LDDA here is merely derived from a coarse modeling of the leakage function – a noised linear transformation of the Hamming weight. The performance difference between the two methods indicates that a more precise estimation of sub-distributions yields better attacking performance for the non-profiling LDDA.

5.3 Experiments for Leaking Set Collision Attack

The third group of experiments mounts the LSCA described in Section 4. Figure 3 shows the hypothesis testing of one LSCA attack using 10,000 to 40,000 traces. The correct subkey hypothesis $k = 108$ gives clear lowest KS distance metric when more than 15,000 traces are used. The distinguishability of the correct subkey increases with the number of traces that are used. Similar to the situation of profiling LDDA, we can observe a drop in the magnitude of the KS distance for the same hypothesis when the number of traces increases. The reason is still the reduction of sampling errors by using more traces.

In addition, we use the provided 100,000 traces to run as many independent experiments as possible for evaluating the LSCA attack. Table 2 summarizes the attacking performance using guessing entropy and t-th order success rate. It can be seen that the LSCA starts a stable success (GE = 1 and 1st order success rate is 100%) with more than 12288 traces, namely, 48 traces per plaintext byte. It is interesting to see that even with a total of 8192 traces (32 traces per plaintext), making 4 guesses still ensures 2/3 success rate. The overall performance is much better than the non-profiling LDDA. However, the comparison with the profiling LDDA shows that the LSCA loses some success rate and requires more traces. The possible reason is that LSCA expands the sampling error. Since the two observed distributions $\mathcal{D}^{x}_{\mathcal{M}}[\Lambda]$ and $\mathcal{D}^{x'}_{\mathcal{M}}[\Lambda]$ are two sampling distributions from the same underlying distribution because of the set collision, the distance $\texttt{Dist}(\mathcal{D}^{x}_{\mathcal{M}}[\Lambda] \| \mathcal{D}^{x'}_{\mathcal{M}}[\Lambda])$ is composed of two components: the distance from $\mathcal{D}^{x}_{\mathcal{M}}[\Lambda]$ to the underlying distribution and the distance from the underlying distribution to $\mathcal{D}^{x'}_{\mathcal{M}}[\Lambda]$. Although the compositional effect is not necessarily as strong as doubling the distance, it is very likely that the sampling error is expanded in the LSCA. While the profiling LDDA only measures one sampling error: the difference between the observed $\mathcal{D}^{x}_{\mathcal{M}}[\Lambda]$ and its underlying distribution. No expansion of sampling error is occurred in the profiling LDDA. Nevertheless, with the slight sacrifice of success rate, the LSCA makes the full use of leakage similarity from the generated the leaking set collisions and therefore does not need assuming profiling capability nor the full control of masks.

6 Conclusion

This work proposes two univariate attacks to overcome the limited protection achieved with the low entropy masking schemes. The first attack—Leakage Distribution Decomposition Attack (LDDA)—reveals the composition of the observed leakage distribution. The second attack—Leaking Set Collision Attack

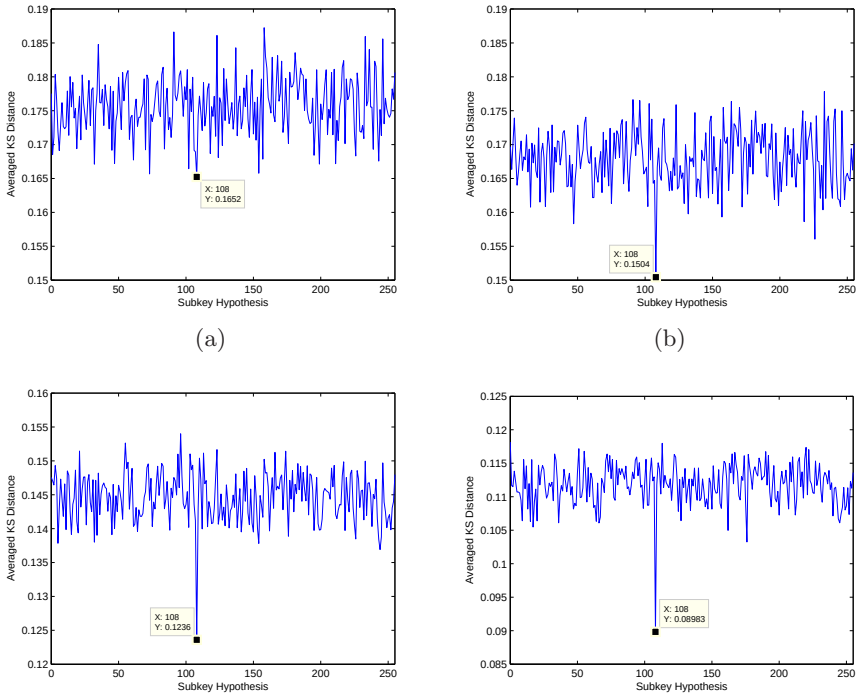


Fig. 3. LSCA hypothesis testing: Kolmogrov-Smirnov distance (y-axis) between observed leakage distributions for the pairings induced from subkey hypothesis (x-axis). Experiments use 10,000 traces (a); 15,000 traces (b); 20,000 traces (c); and 40,000 traces (d).

(LSCA)—extends the concept of side channel collision attacks and does not rely on detailed knowledge of the leakage model or function. Both of the two attacks compare leakage distributions and therefore they have a relatively high requirement on the number of traces. The attacks show that studying a countermeasure with resistance of the first, second or even higher order CPA/DPA is not sufficient to guarantee the resistance to other univariate attacks.

Acknowledgments

We would like to thank the reviewers for the helpful comments. This material is based upon work supported by the National Science Foundation under Grant No. 1261399.

Table 2. LSCA Performance Evaluation

Number of Traces	4096	8192	12288	16384
Guessing Entropy	34.17	5.33	1.00	1.00
1st order Success Rate	0	33.3%	100.0%	100.0%
4th order Success Rate	33.3%	66.7%	100.0%	100.0%

References

1. The dpa contest v4, http://www.dpacontest.org/v4/.
2. S. Bhasin, W. He, S. Guilley, and J.-L. Danger. Exploiting fpga block memories for protected cryptographic implementations. In *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2013 8th International Workshop on*, 2013.
3. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 135–152. Springer Berlin / Heidelberg, 2004.
4. J.-S. Coron and L. Goubin. On boolean and arithmetic masking against differential power analysis. In *Cryptographic Hardware and Embedded Systems CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000.
5. J. Doget, E. Prouff, M. Rivain, and F.-X. Standaert. Univariate side channel attacks and leakage modeling. *Journal of Cryptographic Engineering*, 1:123–144, 2011.
6. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. *Cryptographic Hardware and Embedded Systems–CHES 2008*, pages 426–442, 2008.
7. J. Golic and C. Tymen. Multiplicative masking and power analysis of aes. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003.
8. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 789–789. Springer Berlin / Heidelberg, 1999.
9. K. Lemke-Rust and C. Paar. Analyzing side channel leakage of masked implementations with stochastic methods. In J. Biskup and J. Lopez, editors, *Computer Security ESORICS 2007*, volume 4734 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007.
10. K. Lemke-Rust and C. Paar. Gaussian mixture models for higher-order side channel analysis. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007.
11. A. Moradi, O. Mischke, and T. Eisenbarth. Correlation-enhanced power analysis collision attack. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 125–139. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-15031-9_9.
12. M. Nassar, S. Guilley, and J.-L. Danger. Formal analysis of the entropy / security trade-off in first-order masking countermeasures against side-channel attacks. In

Progress in Cryptology INDOCRYPT 2011, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011.
13. M. Nassar, Y. Souissi, S. Guilley, and J.-L. Danger. Rsm: A small and fast countermeasure for aes, secure against 1st and 2nd-order zero-offset scas. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, 2012.
14. E. Oswald and S. Mangard. Template attacks on masking – resistance is futile. In M. Abe, editor, *Topics in Cryptology CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 243–256. Springer Berlin Heidelberg, 2006.
15. T. Popp and S. Mangard. Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints. In J. Rao and B. Sunar, editors, *CHES 2005*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005.
16. E. Prouff and M. Rivain. A generic method for secure sbox implementation. In S. Kim, M. Yung, and H.-W. Lee, editors, *Information Security Applications*, volume 4867 of *Lecture Notes in Computer Science*, pages 227–244. Springer Berlin Heidelberg, 2007.
17. P. Schaumont and K. Tiri. Masking and dual-rail logic don't add up. In P. Paillier and I. Verbauwhede, editors, *CHES 2007*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007.
18. W. Schindler, K. Lemke, and C. Paar. A stochastic model for differential side channel cryptanalysis. In J. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005.
19. K. Schramm, G. Leander, P. Felke, and C. Paar. A Collision-Attack on AES. In *Cryptographic Hardware and Embedded Systems - CHES 2004*. Springer Berlin / Heidelberg, 2004.
20. K. Schramm and C. Paar. Higher order masking of the aes. In D. Pointcheval, editor, *Topics in Cryptology CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer Berlin Heidelberg, 2006.
21. F.-X. Standaert, T. G. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. *Advances in Cryptology — EUROCRYPT 2009*, pages 443–461, 2009.
22. N. Veyrat-Charvillon and F.-X. Standaert. Mutual information analysis: How, when and why? In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 429–443. Springer Berlin Heidelberg, 2009.
23. C. Whitnall, E. Oswald, and L. Mather. An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis. In E. Prouff, editor, *Smart Card Research and Advanced Applications*, volume 7079 of *Lecture Notes in Computer Science*, pages 234–251. Springer Berlin Heidelberg, 2011.