

Revisiting a Masked Lookup-Table Compression Scheme

Srinivas Vivek^(✉) 

University of Bristol, Bristol, UK
sv.venkatesh@bristol.ac.uk

Abstract. Lookup-table based side-channel countermeasure is the prime choice for masked S-box software implementations at very low orders. To mask an n -bit to m -bit S-box at first- and second- orders, one requires a temporary table in RAM of size $m \cdot 2^n$ bits. Recently, Vadnala (CT-RSA 2017) suggested masked table compression schemes at first- and second-orders to reduce the table size by (approximately) a factor of 2^l , where l is a parameter. Though greater compression results in a greater execution time, these proposals would still be attractive for highly resource constrained devices.

In this work, we contradict the second-order security claim of the second-order table compression scheme by Vadnala. We do this by exhibiting several pairs of intermediate variables that jointly depend on the bits of the secret. Motivated by the fact that randomness is also a costly resource for highly resource constrained devices, we then propose a variant of the first-order table compression scheme of Vadnala that has the new randomness complexity of about l instead of 2^l for the original proposal. We achieve this without inducing any noticeable difference in the overall execution time or memory requirement of the original scheme. Finally, we show that the randomness complexity of l is optimal in an algebraic sense.

Keywords: Side-channel attack · Masking · Block cipher Implementation

1 Introduction

Side-channel attacks on cryptographic implementations exploit physical characteristics of an execution such as timing, power consumption or electromagnetic emission pattern, to name but a few [Koc96, KJJ99]. Block cipher implementations have been a major target for these attacks. Over the years, a number of countermeasures against these attacks have been developed too. Of these, (boolean) *masking* is one of the very first and still a popular technique to protect

This work has been supported in part by the European Union's H2020 Programme under grant agreement number ICT-644209 (HEAT), and by the EPSRC under grant number EP/M016803/1.

block cipher implementations [CJRR99, ISW03]. The basic idea behind masking is to split every sensitive variable into one or more shares and process them in such a way that intermediate variables do not reveal information about these sensitive variables. An implementation is said to be secure against t -th order attacks in the *probing leakage model* if any subset of t intermediate variables (including the input and output shares) jointly is statistically independent of the secret variables [ISW03]. Hence at least $t + 1$ shares are needed for each of the secret inputs to achieve t -th order security. As the number of shares increase so does the complexity of the side-channel attacks [CJRR99, PR13, DDF14]. Moreover, since the processing of affine functions is straightforward, the main challenge in masking block ciphers is the masking of the non-linear operations, in particular, the S-box functions.

Recent years have witnessed an increased focus on the design and improvement of higher-order (boolean/arithmetic) masking schemes for S-boxes. For instance, see [CGPZ16, GR16, PV16, CRZ17, GR17, GRVV17, JS17] and the references within. These masking schemes can roughly be categorised into polynomial/ arithmetic-circuit based masking schemes (including the bit-sliced masking technique) on one hand, and look-up table-based masking schemes on the other. As the above works have shown, at higher orders, polynomial-based schemes have been more efficient than table-based schemes in terms of time, memory and randomness complexity. In spite of the above advancement of polynomial-based masking schemes, at very low orders, such as first- and second-orders, table-based masking schemes are the most effective due to low overheads. Unsurprisingly, vast majority of the commercial implementations opt just for first- or second-order masked implementations due to efficiency concerns.

Original Look-up Table-based Masking. The original table-based first-order masking of an (n, m) -S-box S consists of creating a temporary table $T : \{0, 1\}^n \rightarrow \{0, 1\}^m$ in RAM [CJRR99]:

$$T(a) = S(x_1 \oplus a) \oplus y_1 \quad \forall 0 \leq a \leq 2^n - 1,$$

where x_1 and x_2 are the input shares such that the secret $x = x_1 \oplus x_2 \in \{0, 1\}^n$, and y_1 and y_2 are the output shares such that $S(x) = y_1 \oplus y_2 \in \{0, 1\}^m$. Using T , y_2 can simply be computed as $y_2 = T(x_2)$. The RAM memory requirement for the table T is $m \cdot 2^n$ bits.

Prouff and Rivain [PR07] suggested a first-order S-box masking scheme that mainly requires only two m -bit registers, hence doing away with the need to store the table T . Though this method only requires (essentially) a constant amount of memory, the overhead induced is a factor of about 30–35, while it is just 2 - 3 for the original method [Vad17]. Moreover, in the original method, the table T can be computed “offline” hence significantly reducing the “online” computation time. But in the method of [PR07], the whole table is computed (on the fly) during the online phase and hence the relatively large overhead.

The second-order S-box masking schemes of Schramm and Paar [SP06], and Rivain, Dottax and Prouff [RDP08] also require a temporary table of size $m \cdot 2^n$

bits, while the scheme of Coron [Cor14] requires $3 \cdot m \cdot 2^n$ bits [CRZ17]. The authors of [RDP08] also suggest a second-order scheme that requires only two m -bit registers and 2^n bits of RAM memory.

Masked Table Compression. In order to reduce the RAM memory requirement for highly resource constrained devices, Rao *et al.* [RRST02] suggested a table compression scheme for first-order masking that requires only $\approx m \cdot 2^{n-1}$ bits for the temporary table. For the case of AES, one now needs only 128 bytes for the table instead of 256 bytes for the original table-based method. In general, one can improve the memory complexity of the method of [RRST02] to $\approx m \cdot 2^n / l$ bits, for a parameter l such that $1 \leq l \leq m$.

Inspired by the method of [RRST02], Vadnala [Vad17] suggested masked table compression techniques that achieve better compression for both the first- and second-order S-box masking. It is shown that the memory requirement for the first-order case can be reduced to $\approx m \cdot 2^{n-l} + (n-l) \cdot 2^l$ bits, where l is a parameter, called *compression level*, such that $1 \leq l \leq n$. For the second-order case, it is shown to be $\approx m \cdot 2^{n-l} + (n-l+1) \cdot 2^l$ bits. The author also investigated the (online-)time and (RAM) memory trade-off in between the two extremes mentioned above for the original method. Reasonably efficient first- and second-order masked implementations of AES-128 were obtained using only about 40 bytes of RAM memory [Vad17]. The proposed schemes were argued to be secure in the probing leakage model [ISW03].

Let us very briefly illustrate the technique of [Vad17] for the first-order case. The main idea is to “pack” 2^l table entries of the original randomised table into a single entry of table T_1 :

$$T_1(a^{(1)}) = \left(\bigoplus_{0 \leq i \leq 2^l - 1} S((a^{(1)} \oplus r_i) \parallel i) \right) \oplus y_1, \quad \forall 0 \leq a^{(1)} \leq 2^{n-l} - 1,$$

where $r_i \in \{0, 1\}^{n-l}$ are uniform random and independently sampled. One needs to carefully access this table to produce another table (that need not be stored) which is then securely accessed with the shares of the remaining l -bits of the secret x (cf. Sect. 2.1). Note that for the given compression level l , one needs to make 2^l calls to a random number generator to generate the r_i s.

1.1 Our Contribution

We *contradict* the second-order security claim of the second-order masked table compression scheme(s) of [Vad17]. We exhibit a second-order attack on the scheme(s) by demonstrating the existence of several pairs of intermediate variables that jointly depend on the secret (cf. Lemma 1). Our attack is, in spirit, similar to the third-order attack suggested by Coron, *et al.* [CPR07] on the higher-order masking scheme of Schramm and Paar [SP06].

Motivated by the fact that the generation of quality randomness is possibly a costly operation on highly resource constrained devices, we then revisit the first-order scheme(s) of [Vad17] and propose a variant (first-order) scheme(s)

that requires only about l calls to a random number generator instead of 2^l required for the original scheme(s) (cf. Sect. 2.2). Our main idea is to generate the 2^l values r_i using only $l + 1$ random values $\gamma_0, \dots, \gamma_l \in \{0, 1\}^{n-l}$. We do this by setting r_i to a subset (xor) sum of the γ_j s. Other than this difference, the rest of our method is the same as in [Vad17]. This implies that apart from the difference in the time to compute the r_i s at the very beginning, there is hardly any noticeable difference in the time and memory complexity of our method and that of [Vad17]. We too prove the first-order security of our scheme in the probing leakage model (cf. Theorem 1). It may be noted that it is straightforward to securely compose first-order secure schemes in a bigger construction.

Finally, we show that the randomness complexity of l that we achieved is (nearly) *optimal* in an algebraic sense (cf. Sect. 2.3). Specifically, our computation model assumes that the only arithmetic operations allowed are *xors*, i.e., \mathbb{F}_2 -linear operations. This is a reasonable assumption since nearly all the known table-based masking schemes satisfy this assumption [CJRR99, RRST02, SP06, PR07, RDP08, Cor14, CRZ17, Vad17].

Organisation of the Paper. To gradually introduce the techniques of masked table compression, we first describe our contributions for the first-order case in Sect. 2 before presenting our attack on the second-order scheme in Sect. 3.

2 Improved First-Order Table Compression Scheme

2.1 Original First-Order Scheme

Before we present our improved first-order masked table compression scheme, let us first briefly recollect the original first-order proposal from [Vad17, Sect. 2]. The notation we use here is somewhat different from that in [Vad17, Sect. 2], and we summarise the changes in Remark 1. Throughout the paper, by $b \stackrel{\$}{\leftarrow} \{0, 1\}^k$ we denote a uniform random and independent sampling of a k -bit string.

Consider an (n, m) -S-box $S : \{0, 1\}^n \rightarrow \{0, 1\}^m$, where $n \geq m$. The task is to securely evaluate $S(x)$, given the input shares $x_1 \stackrel{\$}{\leftarrow} \{0, 1\}^n$ and x_2 such that $x = x_1 \oplus x_2 \in \{0, 1\}^n$, ensuring that no intermediate variable is statistically dependent on the “secret” x . The outputs are two shares $y_1 \stackrel{\$}{\leftarrow} \{0, 1\}^m$ and y_2 such that $S(x) = y_1 \oplus y_2 \in \{0, 1\}^m$.

Let $1 \leq l \leq n$ be the compression level. Define the functions $S_i : \{0, 1\}^{n-l} \rightarrow \{0, 1\}^m$ ($0 \leq i \leq 2^l - 1$) as

$$S_i(a^{(1)}) := S(a^{(1)}||i), \quad \forall 0 \leq a^{(1)} \leq 2^{n-l} - 1, \tag{1}$$

where i is represented using l bits. The main idea in [Vad17] is to “pack” 2^l table entries of the original randomised table into a single entry of table T_1 . More precisely, let

$$T_1(a^{(1)}) = \left(\bigoplus_{0 \leq i \leq 2^l - 1} S_i(a^{(1)} \oplus r_i) \right) \oplus y_1, \quad \forall 0 \leq a^{(1)} \leq 2^{n-l} - 1, \tag{2}$$

where

$$r_i \stackrel{\$}{\leftarrow} \{0, 1\}^{n-l}, \quad \forall 0 \leq i \leq 2^l - 1, \quad (3)$$

are uniform random and independently sampled.

Let

$$x =: x^{(1)} \parallel x^{(2)}, \quad \text{where } x^{(1)} \in \{0, 1\}^{n-l}, \quad x^{(2)} \in \{0, 1\}^l. \quad (4)$$

Similarly, let

$$x_1 =: x_1^{(1)} \parallel x_1^{(2)}, \quad \text{where } x_1^{(1)} \in \{0, 1\}^{n-l}, \quad x_1^{(2)} \in \{0, 1\}^l, \quad (5)$$

and

$$x_2 =: x_2^{(1)} \parallel x_2^{(2)}, \quad \text{where } x_2^{(1)} \in \{0, 1\}^{n-l}, \quad x_2^{(2)} \in \{0, 1\}^l. \quad (6)$$

In [Vad17, Sect. 2] it is mentioned that $x_1^{(1)} = \bigoplus_{0 \leq i \leq 2^l - 1} r_i$. But this is not a necessity and we assume that x_1 is independently chosen. The next step is to compute a table $U : \{0, 1\}^l \rightarrow \{0, 1\}^m$ comprising of all the values $S_i(x^{(1)}) \oplus y_1$, where $0 \leq i \leq 2^l - 1$, by securely accessing the tables T_1 and S_i as follows:

$$S_i(x^{(1)}) \oplus y_1 = T_1((x_1^{(1)} \oplus r_i) \oplus x_2^{(1)}) \oplus \bigoplus_{0 \leq j \leq 2^l - 1, j \neq i} S_j(((x_1^{(1)} \oplus r_i) \oplus x_2^{(1)}) \oplus r_j).$$

For security considerations, the expression inside the parentheses above and elsewhere must be evaluated with higher precedence. To compute the second output share $y_2 = S(x) \oplus y_1$, the table U needs to be accessed at $x^{(2)}$. But if one directly accesses the table U as mentioned, then it would leak l -bits of the secret x . Therefore, *instead* of creating the table U , a randomised table T_2 corresponding to U shifted by $x_1^{(2)}$ is created as follows:

$$T_2(a^{(2)}) = T_1((x_1^{(1)} \oplus r_{(a^{(2)} \oplus x_1^{(2)})}) \oplus x_2^{(1)}) \oplus \bigoplus_{0 \leq j \leq 2^l - 1, j \neq (a^{(2)} \oplus x_1^{(2)})} S_j(((x_1^{(1)} \oplus r_{(a^{(2)} \oplus x_1^{(2)})}) \oplus x_2^{(1)}) \oplus r_j), \quad (7)$$

where $0 \leq a^{(2)} \leq 2^l - 1$. Finally, compute

$$y_2 = T_2(x^{(2)}).$$

The above scheme is proven to be first-order secure in the probing leakage model [ISW03]. Namely, every intermediate variable (including the input and output shares) is shown to be independent of the secret x . Note that the table T_1 can be computed offline. As the value of the compression level l increases, then so does the online computation time. The table T_1 has 2^{n-l} m -bit entries, while the table T_2 has 2^l m -bit entries. Hence the combined size of the two

tables is $(2^{n-l} + 2^l) \cdot m$ bits compared to the $2^n \cdot m$ bits needed for the original randomised table-based (first-order) masking scheme. But in the above scheme we now need to also store the 2^l random values r_i each $n - l$ bits long.

It is suggested in [Vad17] how to do away with the need to store the table T_2 . This is based on a first-order S-box masking scheme from [PR07] that mainly uses only two registers (instead of a table) to compute the output shares y_1 and y_2 . The only (implicit) requirement to apply the method of [PR07] in different contexts is that random access must be possible for the table that is being masked. Since the entries of the table T_2 can be computed in any arbitrary order, one can straightforwardly apply the technique of [PR07] in the current context. Hence the RAM memory complexity of the first-order table compression scheme from [Vad17] is approximately $2^{n-l} \cdot m + 2^l \cdot (n - l)$ bits.

Remark 1. The variables $x_1, x_1^{(1)}, x_1^{(2)}, x_2, x_2^{(1)}, x_2^{(2)}, y_1$, and $a^{(1)}$, in this section correspond to, respectively, $r, r^{(1)}, r^{(2)}, x_1, x_1^{(1)}, x_1^{(2)}, s$, and u , in [Vad17, Sect. 2]. The final step that computes y_2 is also slightly different compared to [Vad17].

2.2 Our Method

Our main idea to reduce the randomness complexity of the first-order scheme from [Vad17] is as follows. Instead of choosing 2^l random values $r_i \stackrel{\$}{\leftarrow} \{0, 1\}^{n-l}$ (cf. (3)), we compute the required r_i using only $l + 1$ random values

$$\gamma_j \stackrel{\$}{\leftarrow} \{0, 1\}^{n-l} \quad \forall 0 \leq j \leq l$$

by xoring different subsets of this smaller set of random values. By

$$\text{bits}_l(i) \in \mathbb{F}_2^l \quad \forall 1 \leq i \leq 2^l - 1$$

we mean an l -bit vector consisting of the bits in the binary representation of i . Let $\text{bits}_l(i)[0]$ denote the least significant bit and, consequently, $\text{bits}_l(i)[l - 1]$ denotes the most significant bit of i (which could possibly be 0). Define

$$\begin{aligned} r_0 &:= \gamma_l, \\ r_i &:= \sum_{j=0}^{l-1} \text{bits}_l(i)[j] \cdot \gamma_j, \quad \forall 1 \leq i \leq 2^l - 1. \end{aligned} \tag{8}$$

Hence each of r_1, \dots, r_{2^l-1} is computed as the xor of the subset of γ_j s defined by the binary representation of their indices. When $i = 0$, the subset xor of γ_j s is zero. Hence r_0 is set to a fresh random value. This procedure is summarised in Algorithm 1.

Once the values r_i are generated and stored, then the rest of the procedure is the same as in the original scheme recollected in Sect. 2.1 (but also see Remark 2). Hence the proof of correctness for our improved method follows automatically. For completeness, we summarise the complete (improved) first-order masked table compression scheme in Algorithm 4.

Algorithm 1. Computing $\{r_0, \dots, r_{2^l-1}\}$ according to (8).

```

Input:  $\gamma_0, \dots, \gamma_l \in \{0, 1\}^{n-l}$ .
Output:  $r_0, \dots, r_{2^l-1} \in \{0, 1\}^{n-l}$ .
1:  $r_0 \leftarrow \gamma_l$ 
2: for  $i \leftarrow 1$  to  $2^l - 1$  do
3:    $r_i \leftarrow 0$ 
4:   for  $j \leftarrow 0$  to  $l - 1$  do
5:     if  $\text{bits}_l(i)[j] \neq 0$  then
6:        $r_i \leftarrow r_i \oplus \gamma_j$ 
7:     end if
8:   end for
9: end for
10: return  $r_0, \dots, r_{2^l-1}$ 

```

Algorithm 2. Computing table T_1 for first-order masked table compression (cf. (2)).

```

Input:  $(n, m)$ -S-box table  $S$ , an output share  $y_1 \in \{0, 1\}^m$ ,  $\{r_0, \dots, r_{2^l-1}\}$  from Algorithm 1.
Output: Table  $T_1$ .
1: define  $S_i(a^{(1)}) := S(a^{(1)} || i)$  (cf. (1))
2: for  $a^{(1)} \leftarrow 0$  to  $2^{n-l} - 1$  do
3:    $z \leftarrow y_1$ 
4:   for  $i \leftarrow 0$  to  $2^l - 1$  do
5:      $z \leftarrow z \oplus S_i(a^{(1)} \oplus r_i)$ 
6:   end for
7:    $T_1(a^{(1)}) \leftarrow z$ 
8: end for
9: return  $T_1$ 

```

Remark 2. In Algorithm 3, the variable z is initialised to $T_1(\text{ind}_1)$ and then xored with $\bigoplus S_j(\text{ind}_2)$. But in [Vad17, Sect. 2], the variable z is initialised to 0 and the above xor was computed at the end. The latter approach could lead to a first-order security flaw for our method due to the “random” values r_i being related in our method.

Remark 3. The execution time for our method (Algorithm 4) and that of [Vad17] is the same except for the time required to generate the values r_i . In the latter method it requires 2^l calls to the random number generator, while for our method it needs only $l + 1$ calls plus the computation of $l \cdot 2^{l-1}$ xors.

Remark 4. The RAM memory complexity is the same for both our method and for [Vad17] since the extra variables γ_j in our method can be discarded right after computing and storing the values r_i . The RAM memory complexity for both the methods is approximately $2^{n-l} \cdot m + 2^l \cdot (n - l)$ bits (in spite of computing the table T_2 on the fly).

Algorithm 3. Computing table T_2 for first-order masked table compression (cf. (7)).

Input: (n, m) -S-box table S , two input shares x_1 and x_2 such that $x_1 \oplus x_2 = x \in \{0, 1\}^n$, an output share $y_1 \in \{0, 1\}^m$, $\{r_0, \dots, r_{2^l-1}\}$ from Algorithm 1, table T_1 from Algorithm 2.

Output: Table T_2 .

```

1: define  $S_i(a^{(1)}) := S(a^{(1)}||i)$  (cf. (1)),  $x_1 =: x_1^{(1)} || x_1^{(2)}$  (cf. (5)),  $x_2 =: x_2^{(1)} || x_2^{(2)}$ 
   (cf. (6))
2: for  $a^{(2)} \leftarrow 0$  to  $2^l - 1$  do
3:    $k \leftarrow a^{(2)} \oplus x_1^{(2)}$ 
4:    $ind_1 \leftarrow (x_1^{(1)} \oplus r_k) \oplus x_2^{(1)}$ 
5:    $z \leftarrow T_1[ind_1]$ 
6:   for  $j \leftarrow 0$  to  $2^l - 1$  do
7:     if  $j \neq k$  then
8:        $ind_2 \leftarrow ind_1 \oplus r_j$ 
9:        $z \leftarrow z \oplus S_j(ind_2)$ 
10:    end if
11:  end for
12:   $T_2(a^{(2)}) \leftarrow z$ 
13: end for
14: return  $T_2$ 

```

Remark 5. The randomness complexity of our method in terms of the number of calls to a random number generator is $l + 3$ instead of $2^l + 2$ for [Vad17]. In terms of the number of random bits generated, it is $(l + 1) \cdot (n - l) + n + m$ for ours instead of $2^l \cdot (n - l) + n + m$ for [Vad17].

The above complexity estimates are exclusively for the masked computation of a single S-box and hence does *not* include the processing of the full cipher. We refer to [Vad17, Sect. 4] for a concrete performance evaluation of masked AES-128 on a 32-bit ARM Cortex-M3 based micro-controller. We expect that on such relatively big architectures the execution times for our method and that of [Vad17] will not differ significantly.

Theorem 1. *Algorithm 4 is first-order secure in the probing leakage model.*

Security Proof. To this end, we just need to show that every intermediate variable is independent of the secret input x . It is obvious that all the intermediate values appearing until (including) Step 7 of Algorithm 4 are independent of x since they can be computed offline. These intermediate variables can simply be simulated by picking suitable random values. Out of the intermediate variables occurring in the remaining Steps 8, 9 and 10 of Algorithm 4, the variables $x_2^{(2)} = x^{(2)} \oplus x_1^{(2)}$, y_1 , and $y_2 = T_2(x_2^{(2)}) = S(x) \oplus y_1$ in Steps 9 and 10 are clearly independent of x . This leaves us to deal with only the variables (including the inputs and outputs) occurring in the computation of table T_2 in Algorithm 3.

The (probability distribution of the) variable $k = a^{(2)} \oplus x_1^{(2)}$ is uniform random and independent of x due to x_1 . Since each $r_i \in \{0, 1\}^{n-l}$ ($0 \leq i \leq 2^l - 1$)

Algorithm 4. Improved first-order masked table compression

Input: (n, m) -S-box table S , two input shares x_1 and x_2 such that $x_1 \oplus x_2 = x \in \{0, 1\}^n$.

Output: Two output shares y_1 and y_2 such that $y_1 \oplus y_2 = S(x) \in \{0, 1\}^m$.

- 1: **define** $x_2 =: x_2^{(1)} \parallel x_2^{(2)}$ (cf. (6))
 - 2: $y_1 \leftarrow \{0, 1\}^m$
 - 3: **for** $j \leftarrow 0$ **to** l **do**
 - 4: $\gamma_j \stackrel{\$}{\leftarrow} \{0, 1\}^{n-l}$
 - 5: **end for**
 - 6: Compute $r_0, \dots, r_{2^l-1} \leftarrow$ Algorithm 1 ($\gamma_0, \dots, \gamma_l$)
 - 7: Compute table $T_1 \leftarrow$ Algorithm 2 ($S, y_1, \{r_0, \dots, r_{2^l-1}\}$)
 { All the above steps may be computed offline. }
 - 8: Compute table $T_2 \leftarrow$ Algorithm 3 ($S, x_1, x_2, y_1, \{r_0, \dots, r_{2^l-1}\}, T_1$)
 - 9: $y_2 \leftarrow T_2(x_2^{(2)})$
 - 10: **return** y_1, y_2
-

is uniform random and independent of x and x_1 (because r_i s are xors of uniform random and independent γ_j s), the variables $x_1^{(1)} \oplus r_k$ and $ind_1 = x^{(1)} \oplus r_k$ are uniform random and independent of x . Because each entry of the table T_1 is masked with y_1 , hence they too are uniform random and independent of x . This implies that the initial value of $z = T_1(ind_1)$ is also uniform random and independent of x .

Consider the values assumed by the variable $ind_2 = x^{(1)} \oplus r_k \oplus r_j$. Since $j \neq k$ and if $j, k \neq 0$, it is easy to see that $r_k \oplus r_j = r_{k \oplus j}$. Since all the r_i s are uniform random and independent of x , so is ind_2 . If $j = 0$, then $ind_2 = x^{(1)} \oplus r_k \oplus \gamma_l$, and if $k = 0$, then $ind_2 = x^{(1)} \oplus \gamma_l \oplus r_j$, and the above conclusion follows easily. This also means that $S_j(ind_2)$ occurring in Step 9 of Algorithm 3 is also independent of x . Finally, we need to show that all the values of z from Step 9 (including the Step 12) are independent of x . As reasoned above, the initial value of $z = T_1(ind_1)$ is masked with y_1 . Since each of the values $S_j(ind_2)$ is also independent of y_1 , this implies that all the values assumed by z are always uniform random and independent of x . This completes the security proof. \square

2.3 Lower Bound on Randomness Complexity

We next show that the randomness complexity of our method from Sect. 2.2 has (nearly) optimal randomness complexity in an algebraic sense. Precisely, we prove that one needs to make at least l calls to a random number generator to compute the values $r_0, \dots, r_{2^l-1} \in \{0, 1\}^{n-l}$ used to compute table T_2 (cf. Remark 5). Needless to say, this lower bound is also applicable to the original scheme from [Vad17, Sect. 2]. To prove our lower bound, we assume that the only arithmetic operations performed are xors, i.e., only \mathbb{F}_2 -linear operations, which indeed is a typical scenario for table-based masking schemes.

Theorem 2. *Algorithm 4 needs at least $l \cdot (n - l)$ uniform randomly generated bits to be first-order secure in the probing leakage model if only \mathbb{F}_2 -linear operations are performed.*

Proof. Let us assume that fewer than l calls are made to the random number generator to compute the r_i s. Then we will exhibit an intermediate value that depends on the secret x . Let the generated random values be $\nu_1, \dots, \nu_t \in \{0, 1\}^{n-l}$, where $t \leq l - 1$. In the assumed computation model, all the computed values r_0, \dots, r_{2^l-1} will be of the form

$$r_i = c_i \bigoplus_{1 \leq j \leq t} b_j \cdot \nu_j, \quad \text{where } b_j \in \mathbb{F}_2, c_i \in \{0, 1\}^{n-l},$$

for $0 \leq i \leq 2^l - 1$. This implies that there exist some r_p and r_q ($p \neq q$) such that

$$r_p \oplus r_q = c_p \oplus c_q$$

is a constant. The variable $ind_2 = x^{(1)} \oplus c_p \oplus c_q$ in Step 8 of Algorithm 3 when $a^{(2)} = p$ and $j = q$. Hence this value is correlated with the $n - l$ most significant bits of x . This proves our claim. \square

3 Attack on the Second-Order Masked Table Compression Method of [Vad17]

3.1 Original Second-Order Scheme

Before we present our attack on the second-order masked table compression scheme from [Vad17, Sect. 3], let us first recollect the original scheme. To be consistent with the notation in Sect. 2, we will use a slightly different notation here than that in [Vad17, Sect. 3], and we summarise the changes in Remark 6.

The second-order table compression scheme from [Vad17] is based on the second-order S-box masking scheme from [RDP08, Sect. 3.1]. Consider again an (n, m) -S-box $S : \{0, 1\}^n \rightarrow \{0, 1\}^m$, where $n \geq m$. On input three shares $x_1 \stackrel{\$}{\leftarrow} \{0, 1\}^n$, $x_2 \stackrel{\$}{\leftarrow} \{0, 1\}^n$ and x_3 such that $x = x_1 \oplus x_2 \oplus x_3 \in \{0, 1\}^n$, the task is to compute the three output shares $y_1 \stackrel{\$}{\leftarrow} \{0, 1\}^m$, $y_2 \stackrel{\$}{\leftarrow} \{0, 1\}^m$ and y_3 such that $S(x) = y_1 \oplus y_2 \oplus y_3 \in \{0, 1\}^m$. In order for the scheme to be second-order secure in the probing model, the requirement is that the joint probability distribution of any pair of intermediate variables (including the input and the output shares) is statistically independent of the secret x . At a high level the main technique behind the second-order table compression scheme is similar to that of the first-order compression scheme presented in Sect. 2. First, create a table T_1 that “packs” 2^l randomised S-box values:

$$T_1(b^{(1)}) := \left(\left(\bigoplus_{0 \leq i \leq 2^l-1} S_i(x_3^{(1)} \oplus a^{(1)} \oplus r_i) \right) \oplus y_1 \right) \oplus y_2, \quad (9)$$

where

$$b^{(1)} = a^{(1)} \oplus ((x_1^{(1)} \oplus v^{(1)}) \oplus x_2^{(1)}), \tag{10}$$

for all $0 \leq a^{(1)} \leq 2^{n-l} - 1$, $S_i(a^{(1)}) := S(a^{(1)} || i)$ as in (1), $v^{(1)} \stackrel{\$}{\leftarrow} \{0, 1\}^{n-l}$, and $r_i \stackrel{\$}{\leftarrow} \{0, 1\}^{n-l}$ as in (3).

Let us recollect the notations $x =: x^{(1)} || x^{(2)}$, $x_1 =: x_1^{(1)} || x_1^{(2)}$, $x_2 =: x_2^{(1)} || x_2^{(2)}$ from (4) to (6). Similarly, let $x_3 =: x_3^{(1)} || x_3^{(2)}$. The next step is to compute a table $U : \{0, 1\}^l \rightarrow \{0, 1\}^m$ consisting of the values $S_i(x^{(1)}) \oplus y_1$, where $0 \leq i \leq 2^l - 1$, by carefully accessing the tables T_1 and S_i . We have

$$S_i(x^{(1)}) \oplus y_1 \oplus y_2 = T_1(v^{(1)} \oplus r_i) \oplus \bigoplus_{0 \leq j \leq 2^l - 1, j \neq i} S_j(x^{(1)} \oplus r_i \oplus r_j).$$

Now the final output share $y_3 = S(x) \oplus y_1 \oplus y_2$ can be computed by accessing the table U at $x^{(2)}$. Of course, this cannot be done as $x^{(2)}$ must never be computed explicitly. *Instead* of computing the table U , a second-order masked table T_2 is created that is accessed with the shares of $x^{(2)}$.

Let

$$T_2(b^{(2)}) := T_1(v^{(1)} \oplus r_{(x_3^{(2)} \oplus a^{(2)})}) \oplus \bigoplus_{0 \leq j \leq 2^l - 1, j \neq a^{(2)}} S_{(x_3^{(2)} \oplus j)}(x^{(1)} \oplus r_{(x_3^{(2)} \oplus a^{(2)})} \oplus r_{(x_3^{(2)} \oplus j)}), \tag{11}$$

for all $0 \leq a^{(2)} \leq 2^l - 1$, where

$$b^{(2)} := a^{(2)} \oplus ((x_1^{(2)} \oplus v^{(2)}) \oplus x_2^{(2)}), \tag{12}$$

and $v^{(2)} \stackrel{\$}{\leftarrow} \{0, 1\}^l$. Once the table T_2 is computed, the output share y_3 can simply be computed as

$$y_3 = T_2(v^{(2)}).$$

We will not recollect here the exact details of how the tables T_1 and T_2 are computed since it is not necessary to present our attack. We refer to [Vad17, Algorithm 8] for these details. As observed in [Vad17], it is not necessary to store the table T_2 . Instead, it can be computed “on the fly” by making use of the technique from [RDP08, Algorithm 3].

Remark 6. The variables $x^{(1)}$, $x^{(2)}$, $x_1^{(1)}$, $x_1^{(2)}$, $x_2^{(1)}$, $x_2^{(2)}$, x_3 , $x_3^{(1)}$, $x_3^{(2)}$, y_1 , y_2 , $v^{(1)}$, and $v^{(2)}$, in this section correspond to, respectively, y , b , y_1 , b_1 , y_2 , b_2 , x' , y' , b' , s_1 , s_2 , y_3 , and b_3 , in [Vad17, Sect. 3]. The pairs of variables $(a^{(1)}, b^{(1)})$ and $(a^{(2)}, b^{(2)})$ in our description both correspond, in different contexts, to (a, a') in [Vad17, Sect. 3].

3.2 Our Attack

We now present a second-order security flaw in the second-order masked table compression scheme from [Vad17, Sect. 3]. Our attack is, in spirit, similar to the third-order attack suggested in [CPR07] on the higher-order masking scheme of [SP06]. More precisely, we prove the following lemma that establishes the existence of many pairs of intermediate variables that jointly depend on the bits of the secret x .

Lemma 1. *Let $\beta_1, \beta_2 \in \{0, 1\}^l$. Then*

$$T_2(\beta_1) \oplus T_2(\beta_2) = S_{(\beta_1 \oplus x^{(2)} \oplus v^{(2)})}(x^{(1)}) \oplus S_{(\beta_2 \oplus x^{(2)} \oplus v^{(2)})}(x^{(1)}).$$

Proof. From (11), we have

$$T_2(\beta_1) = T_1(v^{(1)} \oplus r_{(x_3^{(2)} \oplus \alpha_1^{(2)})} \oplus \bigoplus_{0 \leq j \leq 2^l - 1, j \neq \alpha_1^{(2)}} S_{(x_3^{(2)} \oplus j)}(x^{(1)} \oplus r_{(x_3^{(2)} \oplus \alpha_1^{(2)})} \oplus r_{(x_3^{(2)} \oplus j)}), \quad (13)$$

where, from (12),

$$\alpha_1^{(2)} := \beta_1 \oplus ((x_1^{(2)} \oplus v^{(2)}) \oplus x_2^{(2)}). \quad (14)$$

From (9), (10) and (14), we have

$$T_1(v^{(1)} \oplus r_{(x_3^{(2)} \oplus \alpha_1^{(2)})}) = y_1 \oplus y_2 \oplus \bigoplus_{0 \leq j \leq 2^l - 1} S_j(x^{(1)} \oplus r_{\beta_1 \oplus x^{(2)} \oplus v^{(2)}} \oplus r_j).$$

From (14) and by a change of index, we obtain

$$T_1(v^{(1)} \oplus r_{(x_3^{(2)} \oplus \alpha_1^{(2)})}) = y_1 \oplus y_2 \oplus S_{(\beta_1 \oplus x^{(2)} \oplus v^{(2)})}(x^{(1)}) \oplus \bigoplus_{0 \leq j \leq 2^l - 1, j \neq \alpha_1^{(2)}} S_{(x_3^{(2)} \oplus j)}(x^{(1)} \oplus r_{(x_3^{(2)} \oplus \alpha_1^{(2)})} \oplus r_{(x_3^{(2)} \oplus j)}).$$

On substituting the above equation in (13), we get

$$T_2(\beta_1) = S_{(\beta_1 \oplus x^{(2)} \oplus v^{(2)})}(x^{(1)}) \oplus y_1 \oplus y_2.$$

Similarly, we obtain

$$T_2(\beta_2) = S_{(\beta_2 \oplus x^{(2)} \oplus v^{(2)})}(x^{(1)}) \oplus y_1 \oplus y_2.$$

Finally,

$$T_2(\beta_1) \oplus T_2(\beta_2) = S_{(\beta_1 \oplus x^{(2)} \oplus v^{(2)})}(x^{(1)}) \oplus S_{(\beta_2 \oplus x^{(2)} \oplus v^{(2)})}(x^{(1)}).$$

This proves the claim. □

The above result suggests that *every* pair of values in the table T_2 *jointly* depends on $n - l$ bits of the secret x . In particular, if the compression level $l = 1$, this means that each pair of values will jointly “leak” all but one bit of the secret.

Remark 7. Our attack only exploits the values in the table T_2 and not the means by which it is computed. Hence our attack is also applicable to the variant scheme in [Vad17, Sect. 3] where T_2 is not stored but only computed on the fly.

Remark 8. For our attack the compression level can be any value l such that $1 \leq l \leq n - 1$. Note that our attack is not applicable when $l = 0$, which corresponds to the scheme from [RDP08, Sect. 3.1], and when $l = n$. Our attack also does not work for those functions S that depend only on the least significant l bits of its input as this part of the input is randomised. But such functions are of little interest for use as cryptographic S-boxes.

Remark 9. In side-channel experiments one hardly gets the values of intermediate variables without any error. Instead, a noisy function of the bits is observed, for e.g., noisy Hamming weight values. We refer to the techniques in [PRB09, SVO+10] to extract the bits of the secret from the noisy experimental data.

Acknowledgements. We would like to thank Srinivas Karthik and Yan Yan for helpful discussions, and also the anonymous reviewers of INDOCRYPT 2017 for helpful comments.

References

- [CGPZ16] Coron, J.-S., Greuet, A., Prouff, E., Zeitoun, R.: Faster evaluation of sboxes via common shares. In: Gierlichs and Poschmann [GP16], pp. 498–514
- [CJRR99] Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener [Wie99], pp. 398–412
- [Cor14] Coron, J.-S.: Higher order masking of look-up tables. In: Nguyen and Oswald [NO14], pp. 441–458
- [CPR07] Coron, J.-S., Prouff, E., Rivain, M.: Side channel cryptanalysis of a higher order masking scheme. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 28–44. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_3
- [CRZ17] Coron, J.-S., Rondepierre, F., Zeitoun, R.: High order masking of look-up tables with common shares. IACR Cryptology ePrint Archive, 2017:271 (2017)
- [DDF14] Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. In: Nguyen and Oswald [NO14], pp. 423–440
- [GP16] Gierlichs, B., Poschmann, A.Y. (eds.): CHES 2016. LNCS, vol. 9813. Springer, Heidelberg (2016). <https://doi.org/10.1007/978-3-662-53140-2>
- [GR16] Goudarzi, D., Rivain, M.: On the multiplicative complexity of boolean functions and bitsliced higher-order masking. In: Gierlichs and Poschmann [GP16], pp. 457–478

- [GR17] Goudarzi, D., Rivain, M.: How fast can higher-order masking be in software? In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 567–597. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_20
- [GRVV17] Goudarzi, D., Rivain, M., Vergnaud, D., Vivek, S.: Generalized polynomial decomposition for s-boxes with application to side-channel countermeasures. IACR Cryptology ePrint Archive 2017:632 (2017)
- [ISW03] Ishai, Y., Sahai, A., Wagner, D.: Private circuits: securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_27
- [JS17] Journault, A., Standaert, F.-X.: Very high order masking: Efficient implementation and security evaluation. IACR Cryptology ePrint Archive 2017:637 (2017)
- [KJJ99] Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener [Wie99], pp. 388–397
- [Koc96] Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9
- [NO14] Nguyen, P.Q., Oswald, E. (eds.): EUROCRYPT 2014. LNCS, vol. 8441. Springer, Heidelberg (2014). <https://doi.org/10.1007/978-3-642-55220-5>
- [PR07] Prouff, E., Rivain, M.: A generic method for secure SBox implementation. In: Kim, S., Yung, M., Lee, H.-W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 227–244. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77535-5_17
- [PR13] Prouff, E., Rivain, M.: Masking against side-channel attacks: a formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_9
- [PRB09] Prouff, E., Rivain, M., Bevan, R.: Statistical analysis of second order differential power analysis. IEEE Trans. Comput. **58**(6), 799–811 (2009)
- [PV16] Pulkus, J., Vivek, S.: Reducing the number of non-linear multiplications in masking schemes. In: Gierlichs and Poschmann [GP16], pp. 479–497
- [RDP08] Rivain, M., Dottax, E., Prouff, E.: Block ciphers implementations provably secure against second order side channel analysis. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 127–143. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71039-4_8
- [RRST02] Rao, J.R., Rohatgi, P., Scherzer, H., Tinguely, S.: Partitioning attacks: Or how to rapidly clone some GSM cards. In: 2002 IEEE Symposium on Security and Privacy, Berkeley, California, USA, 12–15 May 2002, pp. 31–41. IEEE Computer Society (2002)
- [SP06] Schramm, K., Paar, C.: Higher order masking of the AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 208–225. Springer, Heidelberg (2006). https://doi.org/10.1007/11605805_14
- [SVO+10] Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The world is not enough: another look on second-order DPA. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 112–129. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_7

- [Vad17] Vadnala, P.K.: Time-memory trade-offs for side-channel resistant implementations of block ciphers. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 115–130. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52153-4_7
- [Wie99] Wiener, M. (ed.): CRYPTO 1999. LNCS, vol. 1666. Springer, Heidelberg (1999). <https://doi.org/10.1007/3-540-48405-1>