Vote-Based Attack on High-Order Share-Slicing Masking

David Pokorný, Petr Socha, Martin Novotný
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czech Republic
{david.pokorny, petr.socha, martin.novotny}@fit.cvut.cz

***Abstract*—We present a practical side-channel attack against high-order masked software implementations with the share-slice representation. We derive an exact analytical model of power leakage that captures the joint leakage of all shares within a single register.**

Based on this model, we propose an optimal categorization of power consumption samples and introduce a signal-to-noise ratio (SNR) estimation method. As a replacement for the Pearson χ^2 test, we design a simplified vote-based attack in which each sample casts a vote for a key hypothesis and the most frequent candidate is selected, thereby reducing both computational and memory requirements while maintaining comparable attack performance. Finally, we experimentally validate the proposed model, categorization, and attack on a masked AES implementation.

***Index Terms*—Side-Channel Analysis, Mutual Information Analysis, Share-Slicing Masking, Leakage Model**

I. Introduction

Embedded devices, such as smart cards or cryptocurrency wallets, are widely deployed in security-critical applications and therefore represent attractive targets for adversaries. Beyond attacks on cryptographic algorithms themselves, these devices are often vulnerable to side-channel analysis, which exploits information unintentionally leaked during execution.

Since Kocher's Differential Power Analysis (DPA) [1], a wide range of statistical side-channel attacks has been developed. These attacks typically partition measured traces based on hypotheses about secret data and use a distinguisher to evaluate the resulting distributions. Common distinguishers include difference of means [1], correlation [2], Bayesian classification [3], or Mutual Information Analysis (MIA) [4]. The partitioning of traces is guided by a leakage model function, whose goal is to capture the dependency between secret data and observable physical effects [5].

MIA is a widely used black-box technique that estimates probability density functions of measured samples, commonly using histograms or kernel-based methods [6]. The choice of discretization parameters, such as the number and width of bins of a contingency table, has a significant impact on attack performance [4], [7], especially in noisy settings. While these approaches require little prior knowledge of the device, they may be suboptimal compared to attacks based on more accurate leakage modeling.

To mitigate side-channel leakage, modern countermeasures rely on secret sharing-based masking techniques, including threshold implementations (TI) [8], [9], domain-oriented masking (DOM) [10], and probing security [11], [12]. Masking reduces the dependency between leakage and secret values by randomizing sensitive intermediates. As a result, an attacker must either analyze multiple time points (multivariate attacks) or consider a significantly larger state space, increasing the complexity of the attack.

A. Contribution

This work makes the following contributions. First, in section II, we derive an exact analytical power leakage model for share-sliced Boolean-masked software implementations. In contrast to commonly used black-box approaches, the model explicitly captures the joint leakage of all shares stored within a single register.

Based on this model, we propose in section III an optimal categorization of power consumption traces into a contingency table. A key parameter of the model is the signal-to-noise ratio (SNR), for which we provide an estimation algorithm in subsection III-B that operates directly during the attack.

In section IV, leveraging the derived categorization, we introduce a simplified attack, referred to as the *vote-based attack*, which replaces the Pearson χ^2 test. In this approach, each sample casts a vote for a key hypothesis, and the key with the highest number of votes is selected as the most likely candidate.

Additionally, in subsection IV-A, we show how part of the computational complexity can be traded to memory complexity, resulting in a faster execution of the attack.

Finally, we experimentally validate the proposed leakage model, the optimal categorization and the vote-attack on a share-sliced AES implementation in section V.

II. Share-slicing

The most common method to break the dependency between processed data and secret values is masking based on secret sharing. Secret value s is encoded into a so-called sharing $\mathbf{s} = (s_1, \ldots, s_d)$, which consists of d shares [13]. These

shares (except one for correctness) are sampled from a uniform distribution $\mathrm{Unif}(\{0,1\})$, and it holds that

$$s = \mathrm{Dec}(\mathbf{s}) = \bigoplus_{i=1}^{d} s_i, \tag{1}$$

$$\mathbf{s} = \mathrm{Enc}(s), \tag{2}$$

where $\bigoplus$ is exclusive OR. The function $\mathrm{Enc}(\cdot)$ encodes a secret bit s into a sharing $\mathbf{s}$, while $\mathrm{Dec}(\cdot)$ decodes a sharing back into a single bit. Any set of $d-1$ shares from a sharing is statistically independent of the secret value s.

These sharings are stored in memory (considering a software implementation) in one of the following ways:

- **share-sliced**: entire sharing is stored in one register. With a 32-bit register, one can store up to 32 shares of a single sharing. Transformations (operations on shares) can be implemented using bitwise operations. This approach is generally considered dangerous, as the joint distribution of share consumption leaks information. This breaks the assumption of physical independence, meaning that each gate or operation leaks information independently. Observations made in this article support these claims.
- **bit-slice**: multiple sharings are arranged in a bitwise fashion, such that the i-th register contains the i-th share of each sharing. This technique can be effective for the parallel processing of boolean operations. The bit-slice variant helps to satisfy domain-oriented masking [10].

This article does not analyze the operations directly, but instead focuses on their outcomes, i.e., the values held or moved between registers. For further analysis, we discuss only the share-sliced variant, since our analysis is univariate.

A. Power consumption of sharing

For analysis, we use the standard (approximate) model of CMOS consumption, i.e., each bit switching on a pre-charged bus to zeros consumes a constant amount of power. This model results in power consumption that is linearly proportional to the Hamming weight of the value in the register. It is a well-known approximation of the consumption in multi-bit side-channel analysis.

Operations on shared values, particularly non-linear ones, typically introduce additional fresh randomness and are therefore modeled as transformations of probability distributions rather than deterministic functions. Leakage is modeled as a random variable, typically given by the Hamming weight $\mathrm{HW}(\mathbf{s})$. Each sharing is selected with the same probability (from a uniform distribution), but the leakage distribution is not uniform. Sharings with low or high Hamming weight are selected less often than sharings with an average Hamming weight.

Let $d \geqslant 2$ be a fixed order of masking and $s \in \{0,1\}$ be a secret value encoded into sharing $\mathbf{s} \in S$, where $S = \{0,1\}^d$ is the set of all sharings of value s. The sharing $\mathbf{s}$ is stored in a register using the share-slicing method. Sharing $\mathbf{s}$ is randomly chosen and therefore we can define a probability mass function $P^s : S \to [0,1]$ as:

$$P^s(\mathbf{s}) = \begin{cases} \frac{1}{2^{d-1}}, & \text{if } \mathrm{Dec}(\mathbf{s}) = s, \\ 0, & \text{if } \mathrm{Dec}(\mathbf{s}) \neq s. \end{cases} \tag{3}$$

$\mathcal{P}^0$ and $\mathcal{P}^1$ are uniform distributions over feasible outcomes, i.e. sharings that encode a given secret value. The number of feasible outcomes is 2^{d-1} since we choose $d-1$ random bits and the last bit is fully determined as $s_d = s \oplus \left(\bigoplus_{i=1}^{d-1} s_i\right)$.

The leakage function $\mathcal{L}^s(\mathbf{s}) = \mathrm{HW}(\mathbf{s})$, is a discrete random variable that maps sharings into measurable values for a given secret value s, i.e., the Hamming weight w of the sharing. For vertical side-channel attacks, we are especially interested in power consumption over many sharings that encode the constant secret value. For this purpose, we define probability mass function $\mathcal{D}^s(w)$ over random variable $\mathcal{L}^s$ as:

$$\mathcal{D}^s(w) = P^s(\mathcal{L}^s = w) = \begin{cases} \binom{d}{w}\frac{1}{2^{d-1}}, & \text{if } (w \bmod 2 = s), \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

for $w \in \{0, \ldots, d\}$. The distribution $\mathcal{D}^s(w)$ returns the probability that secret value s will be encoded using exactly w non-zero bits. Fig. 1a shows the distribution for masking order $d = 16$.

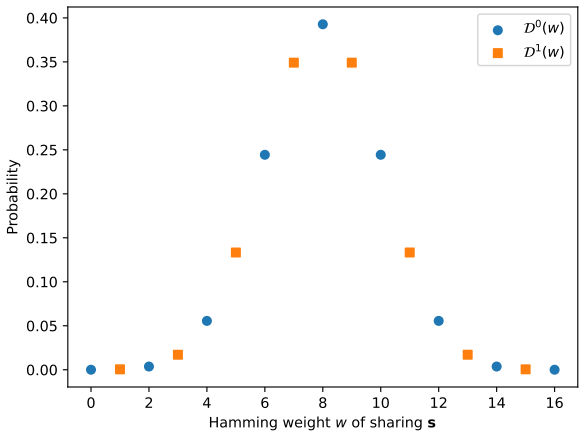
For a more accurate estimation of the leakage function, random noise must be taken into account. In this paper, we assume a constant noise level for a fixed masking order d. Let $\mathcal{N}_{dev}(\mu, \sigma^2)$ denote the normal distribution describing the power consumption of the device, where μ is the average consumption and σ^2 is the variance, representing the noise. The noise resulting from switching the bit of interest will be neglected, i.e. it contributes a constant consumption $c \in \mathbb{R}$. The expected power consumption $x \in \mathbb{R}$ for a given secret value s is a Gaussian mixture:

$$\mathcal{N}^s = \sum_{w=0}^{d} \mathcal{D}^s(w) \cdot \mathcal{N}_{dev}(\mu + cw, \sigma^2). \tag{5}$$

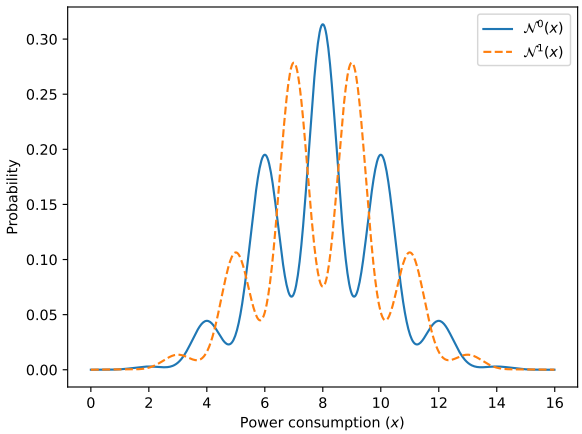
The mean of the device consumption μ is shifted by the Hamming weight of the sharing, multiplied by a constant c that models the consumption caused by switching one bit. This model is plotted in Fig. 1b.

To successfully distinguish the secret values $s = 0$ and $s = 1$, it is sufficient to accurately detect differences between the corresponding distributions. Traces can be partitioned into two groups based on assumptions about the secret value s. When the hypothesis is correct, the resulting power consumption distributions follow the theoretical model and exhibit distinguishable statistical behavior.

Black-box attacks such as MIA typically apply categorical statistical tests, such as the Pearson χ^2 test, using a heuristic, model-agnostic categorization of measured power values. While broadly applicable, this approach does not exploit information provided by a concrete leakage model.



(a) Probability mass distribution $\mathcal{D}^s(w)$ over leakage random variable $\mathcal{L}^s$ for masking order $d = 16$.



(b) Noisy leakage function $\mathcal{N}^s(x)$ for masking order $d = 16$, $c = 1$, $\mu = 0$, $\sigma = 0.5$. The value x is expected power consumption.

Fig. 1: (a) Distribution of Hamming weights for sharing encoding of $s = 0$ and $s = 1$. (b) Corresponding noisy leakage model.

III. Categorization and Model Fitting

We utilize the power model from (5) to derive an optimal categorical partitioning of measured power consumption for Pearson's χ^2 test.

First, we have to fit the observed data and the theoretical model $\mathcal{N}^s(x)$ using standardization. The standardized distribution $\mathcal{Z}^s(x)$ of $\mathcal{N}^s(x)$ for order $d > 2$ is calculated as follows:

$$\mu_w = \frac{c(w - \frac{d}{2})}{\sqrt{\sigma^2 + \frac{c^2 d}{4}}} = \frac{c}{\sigma} \cdot \frac{w - \frac{d}{2}}{\sqrt{1 + (\frac{c}{\sigma})^2 \cdot \frac{d}{4}}}, \tag{6}$$

$$(\sigma')^2 = \frac{\sigma^2}{\sigma^2 + \frac{c^2 d}{4}} = \frac{1}{1 + (\frac{c}{\sigma})^2 \cdot \frac{d}{4}}, \tag{7}$$

$$\mathcal{Z}^s = \sum_{w=0}^{d} \mathcal{D}^s(w) \cdot \mathcal{N}_{dev}(\mu_w, (\sigma')^2). \tag{8}$$

Note that the distribution $\mathcal{Z}^0(x)$ represents the probability of the standardized power consumption x of the device at the time of interest when the secret bit s equals 0. Conversely, $\mathcal{Z}^1(x)$ corresponds to the case where the secret bit $s = 1$.

For fixed masking order d, this standardized model depends only on the power consumption of a single switching bit c and the device noise variance σ^2. After some reformulation, the model is found to depend solely on the signal-to-noise ratio (SNR), expressed as $\frac{c}{\sigma}$. The constant offset μ of the device consumption becomes irrelevant.

For categorical statistical tests, the key points are the intersection points of the standardized leakage distributions $\mathcal{Z}^0(x)$ and $\mathcal{Z}^1(x)$, i.e., the points where $\mathcal{Z}^0(x) = \mathcal{Z}^1(x)$. These intersection points are ideal candidates for defining bin boundaries in a contingency table.

In Fig. 2, we illustrate the $\mathcal{Z}^0(x)$ and $\mathcal{Z}^1(x)$ distributions and their intersection (vertical lines). This figure differs from Fig. 1b only by the standardization and the inclusion of vertical lines for categorization.

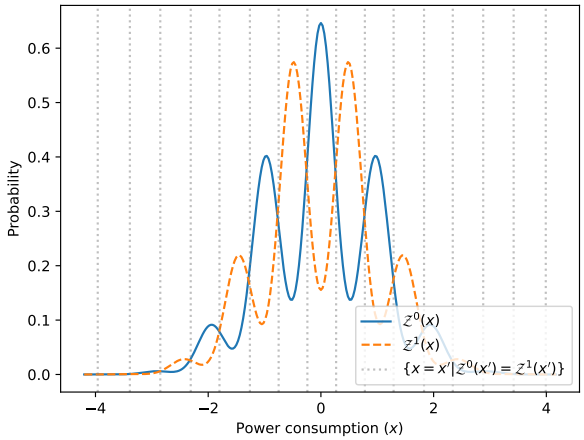


Fig. 2: Standardized leakage distributions $\mathcal{Z}^0(x)$ and $\mathcal{Z}^1(x)$ for masking order $d = 16$ and SNR $\frac{c}{\sigma} = 2$. The intersections (indicated by vertical lines) define the bin boundaries used in categorical statistical tests.

In standardized distributions $\mathcal{Z}^s(x)$, the higher SNR, the more separated the distributions become, and the peaks more pronounced, resulting in more reliable classification.

A. Analyzing Signal-to-Noise Ratio

To ensure that the consumption model accurately reflects the expected power consumption of the device, we must estimate the signal-to-noise ratio (SNR), defined as $\frac{c}{\sigma}$.

The SNR is estimated by fitting the measured standardized power distribution against the theoretical unconditional distribution $\mathcal{Z}(x)$. The latter is modeled as a mixture of

Algorithm 1 Curve Length Difference

Input: Standardized data $x_{m,t}$, where t is the time index and m is the trace index
Output: Estimated SNR for each time t
for all t **do**
    Let n be the number of samples at time t
    Extract distinct values u_i from samples at time t
    Compute occurrences c_i of each value u_i
    Compute interval widths $a_i = \frac{u_{i+1} - u_{i-1}}{2}$
(boundaries handled separately)
    Compute relative frequencies $f_i = \frac{c_i}{n \cdot a_i}$
    Compute squared value differences
$(\Delta u_i)^2 = (u_{i+1} - u_i)^2$
    Compute frequency differences $\Delta f_i = f_{i+1} - f_i$
    Compute empirical curve length
$L_{\text{emp}} = \sum_i \sqrt{(\Delta u_i)^2 + (\Delta f_i)^2}$
    Initialize an empty dictionary `metric`
    for all SNR **do**
        Compute theoretical probabilities $p_i = \mathcal{Z}(u_i)$
        Compute probability differences $\Delta p_i = p_{i+1} - p_i$
        Compute theoretical curve length
$L_{\text{the}} = \sum_i \sqrt{(\Delta u_i)^2 + (\Delta p_i)^2}$
        Store $|L_{\text{emp}} - L_{\text{the}}|$ in `metric`[SNR]
    end for
    Select the SNR with minimal value in `metric` for time t
end for

the conditional distributions $\mathcal{Z}^s(x)$ corresponding to the two secret values:

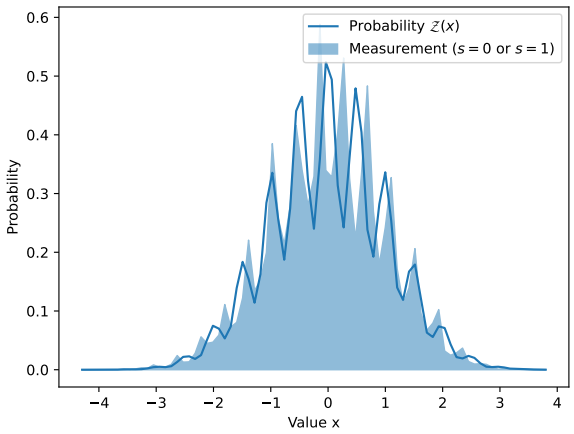
$$\mathcal{Z}(x) = \tfrac{1}{2}\left(\mathcal{Z}^0(x) + \mathcal{Z}^1(x)\right). \tag{9}$$

This model corresponds to a balanced target bit, i.e., the secret values $s = 0$ and $s = 1$ occur with equal probability, an assumption often satisfied in practice due to uniformly distributed inputs such as plaintexts. If the target bit is not balanced, (9) can be modified accordingly to account for the expected distribution.

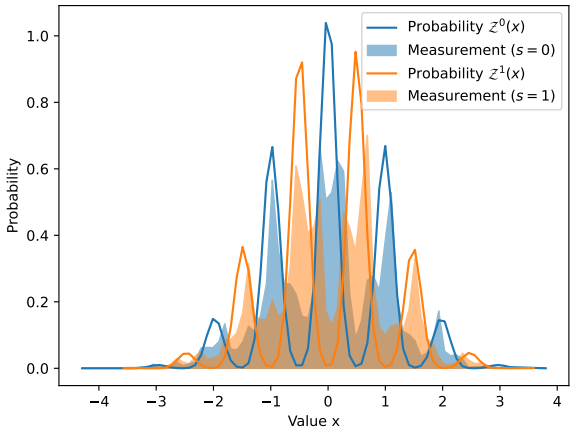
B. Our estimation of SNR

To estimate the SNR, we compare the measured leakage distribution with the theoretical distribution $\mathcal{Z}(x)$ by evaluating the difference in their curve length. Since the SNR primarily affects the prominence of distribution peaks, this difference provides a sensitive estimation criterion. The procedure is summarized in Alg. 1.

For each time point t, the empirical distribution constructed from the standardized measurements is converted to a density estimate in order to compare the discrete measurements with the continuous distribution $\mathcal{Z}(x)$. The resulting estimate is then compared to $\mathcal{Z}(x)$ for a range of candidate SNR values using the curve length difference, where the curve length is computed from Euclidean distances between consecutive samples. The SNR minimizing this difference is then selected.



(a) Empirical distribution of measured data compared to the (best fitting) theoretical model $\mathcal{Z}(x)$.



(b) Data grouped according to the secret value s, with $s = 0$ and $s = 1$.

Fig. 3: Comparison of measured data (standardized and fitted to a continuous distribution) for masking order $d = 16$ and SNR $\frac{c}{\sigma} = 3.4$, selected using the proposed metric. The graph is evaluated only at discrete points, as specified in Alg. 1.

This metric was chosen because the peaks of the unconditional leakage distribution may not be perfectly aligned with those predicted by the theoretical model, for example due to asymmetric peak shapes. This effect is likely due to a linear approximation of the register's power consumption as a function of the number of switching bits. In Fig. 3a, this effect is visible, whereas Fig. 3b shows that this discrepancy disappears after partitioning according to the secret value. The curve length difference does not require explicit alignment of mixture components and therefore provides a more reliable SNR estimate.

For SNR estimation, we compare whether two distributions are equal, an operation that the attack itself is designed to avoid. The goal is to estimate the SNR from a small subset of

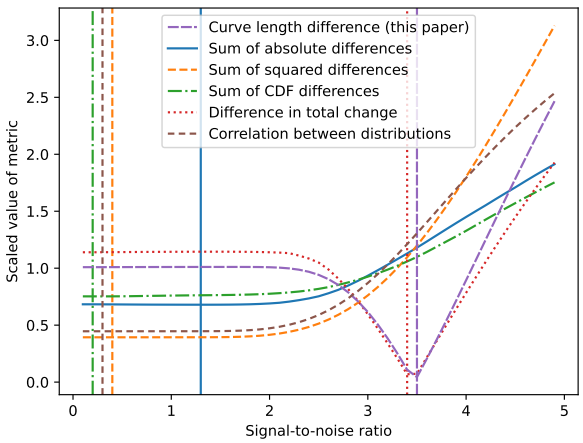


Fig. 4: Comparison of several metrics used to estimate the optimal signal-to-noise ratio (SNR).

measurements, where the estimate itself does not need to be highly precise. Small differences between nearby SNR values have only a limited impact on the resulting classification. A more detailed investigation across different noise levels is left for future work.

C. Comparison of Different SNR Estimations

We evaluated several metrics for estimating the signal-to-noise ratio (SNR) by comparing the empirical leakage distribution with the theoretical model, enabling a quantitative comparison and assessment of their practical suitability. The tested metrics include sum of absolute differences, sum of squared differences, differences of cumulative distribution functions (CDFs), correlation, difference in total change, and the curve length difference proposed in this paper.

Fig. 4 compares the behavior of these metrics over a range of SNR values. Each curve shows a rescaled metric value. Vertical dashed lines indicate the SNR value at which each metric reaches its minimum, representing the estimated optimal SNR. The reference SNR value ($\frac{c}{\sigma} = 3.2$) was obtained using the correct key. The masking order is $d = 16$, and 2,000 traces were used in this experiment.

Metrics based on correlation or pointwise differences consistently underestimate the true SNR. The difference in total change, while computationally simpler, tends to overestimate the SNR in some cases.

Overall, the curve length difference proposed in this paper provides the most reliable estimates across experiments and is therefore used as the default SNR estimation method.

IV. Vote Attack

Given the consumption model $\mathcal{Z}(x)$ and the estimated SNR, we discretize the power consumption into bins for the contingency table. From this categorization, shown as vertical lines in Fig. 2, we can observe that the categories can be grouped into two alternating groups, allowing us to combine even and odd categories. If the observed distribution is known, we only need to store which of these two groups of categories a sample belongs to.

For subsequent analysis, we formulate a hypothesis about the key, which we use to divide the samples into two sets (categories) based on the expected observed distribution. If the key hypothesis is correct, the observations correlate with the predicted category; otherwise, the predicted category appears random.

Let's define two observable categories $\mathcal{C}_0$ for secret $s = 0$ and $\mathcal{C}_1$ for $s = 1$:

$$\mathcal{C}_0 = \{x | \mathcal{Z}^0(x) > \mathcal{Z}^1(x), x \in \mathbb{R}\}, \tag{10}$$

$$\mathcal{C}_1 = \{x | \mathcal{Z}^0(x) \leqslant \mathcal{Z}^1(x), x \in \mathbb{R}\}. \tag{11}$$

Next, we define a test that returns the index of the category based on the measured normalized value x:

$$\mathcal{C}(x) = \begin{cases} 0, & x \in \mathcal{C}_0, \\ 1, & x \in \mathcal{C}_1. \end{cases} \tag{12}$$

As an example, we attack AES first-round S-box under uniformly distributed plaintexts. Let $\mathcal{H}(p, k) = s$ be a hypothesis function, defined as:

$$\mathcal{H}(p, k) = (\text{SBOX}(p \text{ XOR } k) \text{ AND } 0\text{x}1), \tag{13}$$

where p and k denote the n-th byte of the plaintext and key, respectively. This function predicts the bit expected to be encoded using secret sharing and processed by the device. The function $\mathcal{H}(p, k)$ returns an element of $\{0, 1\}$, representing the predicted category of s.

The attack itself compares whether the observed category matches the predicted category, expressed as $\mathcal{C}(x) = \mathcal{H}(p, k)$. Since this is a univariate attack, we use the measured value from only a single specific time t. Let $\mathbf{x}$ represent the measurement, and $x_{m,t}$ denote the sample at time t from the m-th trace. Each vector of samples in a particular time is standardized, meaning that for each t the vector $x_{:,t}$ has a zero expected value and a variance of one. The symbol : means over all possible values. The correct mean and variance used for standardization are computed during the SNR estimation process.

The attack is computed using the following equation for each (sub)key k and every time t:

$$R_{t,k} = \sum_{\text{for all } m} (-1)^{\mathcal{C}(x_{m,t}) \text{ XOR } \mathcal{H}(p,k)}, \tag{14}$$

where $R_{t,k}$ is a 2-dimensional table for storing the results, where the first dimension corresponds to the time t and the second corresponds to the subkey k.

For the correct key, it holds that $\mathcal{C}(x_{m,t})$ and $\mathcal{H}(p, k)$ correlate, thus, across the sum, the exponents are more likely to be zero and less likely to be one (or vice versa in the case of anticorrelation), and the sum diverges. For an incorrect key, the prediction is independent of the key and exponents are equally likely to be zero and one. In this case, the mean value of the sum is around zero.

The correct key therefore corresponds to the absolute maximum value of the sum. Using the absolute value ensures independence from the categorization (e.g., swapping odd and even categories) and from the sign of bit consumption (whether swapping a bit results in higher or lower power consumption). Obtaining the result is done by

$$\{t', k'\} = \text{argmax}(\text{abs}(R)), \tag{15}$$

where abs is the absolute value (element wise) and argmax returns the index of the maximal value. The correct key is k' and the strongest leak occurs in time t'.

A. Acceleration through Precomputation

For faster computation, we precompute the summand (i.e., $(-1)^{\mathcal{C}(x_{m,t}) \text{ XOR } \mathcal{H}(p,k)}$) in (14) and store the results in a look-up table. This table contains vectors corresponding to all key hypotheses for all plaintexts (assuming that the number of subkey–plaintext combinations remains manageable; in the case of AES, we consider 65,536 combinations). This approach eliminates the need to iterate over all possible keys, thereby reducing the number of cycles and intermediate calculations, and replacing them with vector additions.

We can also introduce another variable in the look-up table to account for all sets of subkeys (in the case of AES-128, we attack 16 bytes, not just one byte). However, we will omit this extension in this article for simplicity.

Let $T_{s,p,k}$ be a 3-dimensional look-up table with the following indices: $s \in \{0, 1\}$ for the measured vote (denoted as $\mathcal{C}(x_{m,t})$), p for plaintext, and k for subkeys. The stored value of the table will be -1 or 1, defined as:

$$T_{s,p,k} := \begin{cases} 1 & s = \mathcal{H}(p,k) \\ -1 & s \neq \mathcal{H}(p,k) \end{cases} \tag{16}$$

After each measurement, we compute the votes, i.e. $\mathcal{C}(x_{m,t})$, for each sample and simply add the corresponding part, i.e. $T_{s,p}$, of the look-up table to the result table R defined above.

At the beginning of the attack, $R_{t,k}$ is filled with zeroes. Then for each standardized sample, i.e. $x_{m,t}$, we compute vote $s = \mathcal{C}(x_{m,t})$. With the knowledge of used plaintext p, we add part of look-up table $T_{s,p}$ to the result table R_t. Note that we are adding a vector over subkey k. The addition of vectors can be parallelized or the vector/SIMD operations can be used. In terms of mathematical notation, we compute the following for each sample $x_{m,t}$ with plaintext p:

$$s = \mathcal{C}(x_{m,t}) \tag{17}$$

$$R_t \leftarrow R_t + T_{s,p} \tag{18}$$

This addition has to be done over all samples. Bins of $\mathcal{C}(x_{m,t})$ can be also precomputed.

Another optimization, that can be useful in certain cases, is to avoid immediate addition of the look-up table. Instead, we can count how many times the table would be added and, at the end of the measurement, multiply the table by that count. In this case, the table T does not need to be precomputed, but we need to use a 3-dimensional variable $C_{t,s,p}$ to keep track of the number of accumulations. For each sample $x_{m,t}$ we compute:

$$s = \mathcal{C}(x_{m,t}) \tag{19}$$

$$C_{t,s,p} \leftarrow C_{t,s,p} + 1, \tag{20}$$

and the final result is obtained for each t:

$$R_t = \sum_{\substack{s \in S \\ p \in P}} C_{t,s,p} \cdot T_{s,p}, \tag{21}$$

where $S = \{0, 1\}$ and P is the set of possible plaintexts (relevant to the set of subkeys). Note that $C_{t,s,p}$ is an integer and $T_{s,p}$ is an array over k. The advantage lies in faster data processing, as we only increment a counter for each sample instead of adding an entire array over all subkeys. The drawback, however, lies in increased memory usage.

V. Measurement and Evaluation

For the measurements, we used the ChipWhisperer-Lite platform with a 32-bit STM32F303 microcontroller (ARM Cortex-M4) as the target. The platform features an integrated 10-bit ADC optimized for power measurements [14].

Using the method proposed in [11], we implemented the AES S-box and reformatted the output into a share-sliced representation. We focus on attacking one secret bit as defined in (13), which is encoded using secret sharing in the share-sliced implementation, as described in (1). The plaintext is known and uniformly sampled. Based on hypotheses about the key, we attempt to reveal the correct key and compare it to heuristic categorization.

Fig. 5 shows a histogram representing the observed frequencies used to construct the contingency table for testing a key hypothesis using the Pearson χ^2 test. The binning follows the categorization method proposed in this work. The masking order $d = 16$, and the SNR $\frac{c}{\sigma} = 3.2$. A total of 20,000 traces was used. Such a high number of traces was chosen solely for visualization purposes.

The histogram reveals a clear duality of categories (even versus odd), which is subsequently exploited by the proposed vote-based attack. At the same time, the relatively large number of distinct samples within each category enables the black-box approach based on the Pearson χ^2 test to perform effectively.

A. Categorization evaluation

We compare the proposed categorization method with common heuristic binning strategies used in combination with the Pearson χ^2 test. Fig. 6 shows the probability of recovering the correct key as a function of the number of traces. The remaining curves correspond to standard heuristics available in the NumPy library. The "vote-attack" curve in the figure is discussed in subsection V-B.

The analysis was performed for masking orders 8, 16, and 32. In all three cases, the results were comparable to those obtained using the best heuristic approach, namely square-root binning, where the number of bins is chosen as $\lceil \sqrt{n} \rceil$,

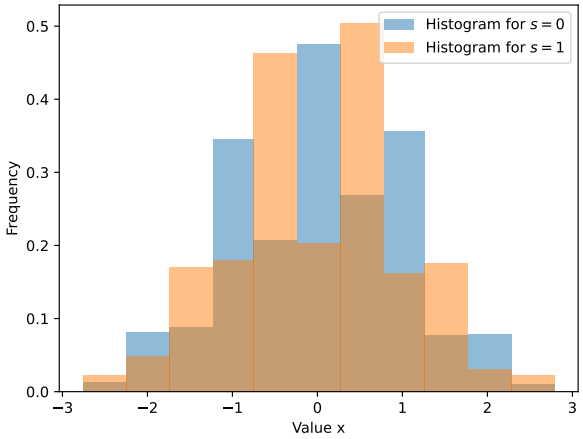


Fig. 5: Histogram of measured power consumption using the proposed categorization.

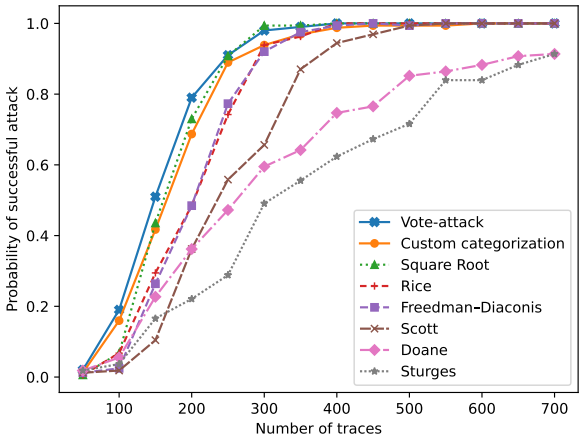


Fig. 6: Comparison of the χ^2 test with different categorization (binning) strategies and the Vote-attack method. The custom categorization corresponds to the approach described in section III, while the remaining binning strategies are common heuristics. Vote-attack corresponds to the attack described in subsection V-B. The x-axis represents the number of traces used for the analysis, and the y-axis shows the probability of correctly recovering the key (out of 256 possibilities). The masking order is $d = 16$.

with n denoting the total number of samples. The proposed categorization does not significantly outperform square-root binning, which achieves comparable results without parameter estimation.

B. Vote-attack evaluation

We further evaluate the vote-based attack described in section IV, employing the custom categorization introduced in section III.

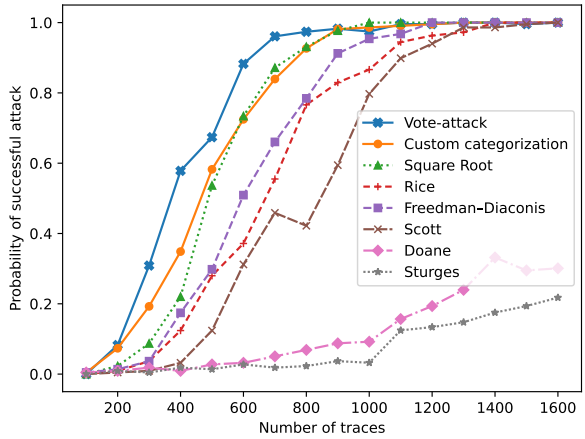


Fig. 7: Comparison of the χ^2 tests with different binning strategies and the vote-based attack for masking order $d = 32$.

The vote-based attack is compared against a black-box approach under the same experimental conditions as in the categorization evaluation. The results are presented in Fig. 6, labeled as *vote-attack*. For all considered masking orders ($d \in \{8, 16, 32\}$), the success rate of the vote-based attack is consistently slightly higher than that of the Pearson χ^2 test with square-root categorization. For completeness, we also include an analysis for masking order $d = 32$ in Fig. 7.

The vote-based attack requires an initial estimation of the signal-to-noise ratio (SNR) in order to define appropriate categorization bins. This estimation, described in section III, can be interpreted as a profiling phase; however, it is performed within the attack itself, as it relies on the same data used for the attack.

The computation of SNR requires a set of pre-measured traces, which serve for normalization of the measurements by providing the mean value and standard deviation for each time t. We analyze how the number of traces used for SNR estimation affects the overall attack performance.

In Fig. 8, we depict the success rate of the attack for different masking orders. The x-axis represents the number of traces used to estimate the normalization parameters (referred to as the *first batch*). For example, when the first batch consists of 100 traces out of 500, the attack first processes 100 traces to compute the mean and standard deviation, standardizes them, and uses them for evaluation of SNR. Subsequently, the remaining 400 traces are standardized using the same parameters.

This approach implies that only the first batch needs to be stored in memory. Once all traces from the first batch are processed, they can be used for voting and discarded. Any subsequent traces are processed on-the-fly and do not require long-term storage.

In Fig. 8, we observe that the size of the first batch has a significant impact on the success of the attack. At the same time, even a relatively small number of traces is sufficient to

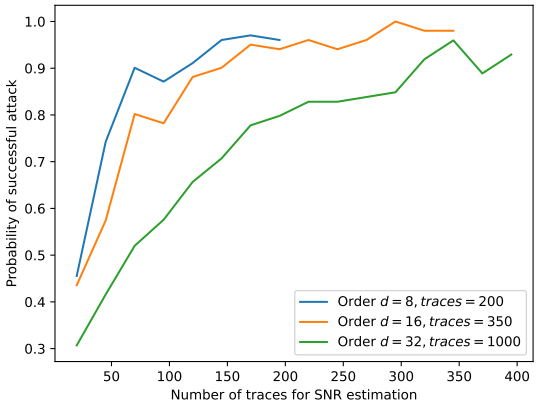


Fig. 8: Effect of first batch size on the success rate of the vote-based attack. The x-axis shows only the first batch used for SNR estimation and bin-boundary computation; the total number of traces is fixed to 200 for $d = 8$, 350 for $d = 16$, and 1000 for $d = 32$.

obtain a good estimate of the mean and standard deviation. For higher noise levels, however, a substantially larger number of power traces is required for a successful attack.

VI. Conclusion

In this work, we analyzed side-channel leakage of a software implementation using share-slicing on a 32-bit architecture. We introduced a theoretical model for the distribution of power consumption under a masking technique based on secret sharing, and demonstrated how this model enables the construction of an optimal categorization for statistical attacks.

We compare the proposed optimal categorization with commonly used heuristic approaches and experimentally demonstrate that the derived categorization performs at least as well as the best evaluated heuristic, namely the square-root method.

Leveraging knowledge of the exact categorization, we derive a novel attack referred to as the *vote-based attack*. In contrast to black-box methods that rely on heuristic categorization to construct contingency tables for the Pearson χ^2 test, our approach exploits the precise leakage model. This enables the design of an attack that requires only two categories and reduces the memory requirement at each time point to a single counter for each key. As a result, the method has lower memory demands and relies only on simple vector operations, leading to improved computational efficiency.

Furthermore, we demonstrate how part of the computational complexity can be shifted to memory. Specifically, for each sample, a unit increment is added to a matrix that accumulates vote counts (see subsection IV-A). This eliminates the need to perform separate computations for each key hypothesis.

When comparing the vote-based attack with heuristic approaches, we observe higher efficiency, i.e., fewer power traces are required. However, the improvement in efficiency is not particularly pronounced. For example, for masking order $d = 16$, the gain corresponds to only a few traces out of approximately 400, while for $d = 32$, the reduction is about 50 traces out of 1000 in total.

Therefore, the main significance of this approach lies in its potential for future research, particularly in extending it to bitsliced implementations, where memory and computational complexity become critical factors. In future work, it is also necessary to investigate the impact of different noise levels on this method.

Our results confirm that even high-order masked implementations can be effectively attacked when sensitive intermediate values are combined in a shared-slice manner.

Acknowledgment

This work has been done within The Advanced Chip Design and Research Center (ACDRC) project focused on innovation and collaboration in semiconductor research and education between Taiwan and the Czech Republic. It was also supported by the Czech Technical University (CTU) grant No. SGS26/184/OHK3/3T/18 “Intelligent, safe, and secure embedded systems”.

AI-Assisted Language Editing

Generative AI tools were used to improve the language and readability of the manuscript. All scientific content and conclusions remain the responsibility of the authors.

References

[1] P. Kocher, J. Jaffe, and B. Jun, *Differential Power Analysis*. Springer Berlin Heidelberg, 1999, pp. 388–397.

[2] E. Brier, C. Clavier, and F. Olivier, *Correlation Power Analysis with a Leakage Model*. Springer Berlin Heidelberg, 2004, pp. 16–29.

[3] S. Chari, J. R. Rao, and P. Rohatgi, *Template Attacks*. Springer Berlin Heidelberg, 2003, pp. 13–28.

[4] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, *Mutual Information Analysis*. Springer Berlin Heidelberg, pp. 426–442.

[5] J. Doget, E. Prouff, M. Rivain, and F.-X. Standaert, “Univariate side channel attacks and leakage modeling,” *Journal of Cryptographic Engineering*, vol. 1, no. 2, pp. 123–144, Aug. 2011.

[6] T. Schneider, A. Moradi, F.-X. Standaert, and T. Güneysu, *Bridging the Gap: Advanced Tools for Side-Channel Leakage Estimation Beyond Gaussian Templates and Histograms*. Springer International Publishing, 2017, pp. 58–78.

[7] F. Durvaux, F.-X. Standaert, and N. Veyrat-Charvillon, *How to Certify the Leakage of a Chip?* Springer Berlin Heidelberg, 2014, pp. 459–476.

[8] S. Nikova, C. Rechberger, and V. Rijmen, *Threshold Implementations Against Side-Channel Attacks and Glitches*. Springer Berlin Heidelberg, 2006, pp. 529–545.

[9] S. Nikova, V. Rijmen, and M. Schläffer, “Secure hardware implementation of nonlinear functions in the presence of glitches,” *Journal of Cryptology*, vol. 24, no. 2, pp. 292–321, Oct. 2010.

[10] H. Gross, S. Mangard, and T. Korak, “Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order,” in *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*, ser. CCS’16. ACM, Oct. 2016.

[11] M. Rivain and E. Prouff, *Provably Secure Higher-Order Masking of AES*. Springer Berlin Heidelberg, 2010, pp. 413–427.

[12] G. Cassiers and F.-X. Standaert, “Trivially and efficiently composing masked gadgets with probe isolating non-interference,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2542–2555, 2020.

[13] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[14] C. O’Flynn and Z. Chen, “Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection,” *Journal of Cryptographic Engineering*, vol. 5, no. 1, pp. 53–69, Nov. 2014.