Blind Trace-Only Segmentation of Cipher Implementations Without Algorithm Metadata

Hyunjun Kim[1][0000−0001−6757−6109], Hwajeong Seo[2][0000−0003−0069−9061], and Anupam Chattopadhyay[1][0000−0002−8818−6983]‡

[1] School of Computer Science and Engineering, Nanyang Technological University, Singapore

[2] Division of IT Convergence Engineering, Hansung University, Seoul, South Korea

khj930704@gmail.com, hwajeong84@gmail.com, anupam@ntu.edu.sg

Abstract. Blind side-channel analysis (BSCA) can infer keys without known inputs or outputs, but practical use still needs an upstream step that locates repeated computation and candidate points of interest in an unlabeled trace. We address this trace-only structuring problem with a two-stage method that uses only the per-sample mean and standard deviation, without algorithm labels or metadata. Stage 1 estimates a repetition scale, start phase, anchor-supported stable core, and period candidates from rank-combined self-similarity. Stage 2 stacks the stable core into a representative repetition and partitions it into relative high- and low-score segments. On 16 block cipher implementations across STM32F303 and XMEGA, the method forms consistent repetition windows in most cases. Post-hoc source and assembly comparison separates exact or edge-inclusive count matches from grouped, microperiod, and ambiguous hierarchy relations, while Top-5 candidates often retain body-related hierarchy. In a representative AES/XMEGA case, the trace-only high-score segments cover the strongest S-box CPA hotspots, indicating that the produced coordinates can prioritize, rather than determine, candidate regions for later CPA or BSCA.

Keywords: Power analysis · Blind side-channel analysis · Trace structuring · Self-similarity analysis · Trace alignment · Points of interest · Block cipher implementations.

1 Introduction

Side-channel analysis (SCA) exploits physical leakage such as power, electromagnetic emanation, or timing to infer implementation-dependent secret state. In ordinary non-profiled SCA, the attacker uses known plaintext or ciphertext, enumerates key candidates, and tests a leakage model at selected points of interest (POIs). The quality of the POI selection and the alignment of repeated operations therefore strongly affect attack cost and success.

‡ Corresponding author.

Blind side-channel analysis (BSCA) aims to reduce or remove the plaintext/ciphertext requirement by exploiting relationships among internal values. It has progressed from joint-distribution attacks to belief propagation, multivariate leakage, deep learning, and multi-POI methods, and has been applied beyond AES to AEAD, PQC, and ARX targets [1,2,3,4,8,9,10,11]. Yet many BSCA procedures still begin after the trace has already been localized: the target time region, repetition structure, POI candidates, or correspondence between repeated operations is assumed or obtained through profiling. In a realistic measurement setting, source code, assembly, execution markers, communication data, and reliable round boundaries may not be available together.

This paper addresses the missing upstream step as the *POI- and structure-blind trace structuring problem*. Given only measured traces and no plaintext, ciphertext, key, source code, assembly, execution markers, or known POIs, the goal is to estimate a repetition scale, start phase, stable repetition core, and intra-repetition candidate segments that can guide later CPA or BSCA. We use only the per-sample mean and standard deviation over traces. Stage 1 ranks self-similar lags across four summary-statistic features, aligns candidate repetition windows, and extracts the longest anchor-supported stable core. Stage 2 stacks that core into a representative repetition and partitions its relative time axis into high- and low-score segments. The result is a hierarchical coordinate system consisting of repetition index and intra-repetition relative position.

Contributions. First, we formulate the trace-structuring stage that must precede key inference under a POI- and structure-blind condition. Second, we propose a two-stage trace-only method that derives repetition-window structure and internal segments from mean/std statistics alone. Third, using 16 block-cipher implementations on STM32F303 and XMEGA, we evaluate whether the trace-only output corresponds to visible implementation-level repetition hierarchies, using source code, assembly, and known-key CPA only as post-hoc interpretation tools.

Section 2 reviews related work and positions this study. Section 3 describes the proposed two-stage trace-only structuring technique, and Section 4 presents experimental results on 16 implementations across two platforms. Section 5 concludes and discusses future work.

2 Related Work

BSCA studies have substantially weakened the input/output assumptions of classical SCA. Joint-distribution key inference [1], multi-round belief propagation [2], multivariate attacks on masked implementations [4], blind SIFA and AEAD attacks [7,8], PQC attacks [9], deep-learning-based multi-POI inference [10], and ARX-oriented attacks [11] all show that key inference can proceed with less direct use of plaintext or ciphertext. However, these works usually rely on some prior localization: leakage windows, candidate POIs, algorithm structure, or temporal correspondence between repeated operations are known, profiled, or manually

Table 1. Input assumptions and output objectives of nearby methods.

Method	Input information	Prior structure	Output objective
SCATTER [5]	Trace, plaintext, key candidate, window	Fixed target and intermediate; alignment is relaxed	Distinguish key candidates
Generic SCARE [6]	Trace, input, output	Constant-time 8-bit execution with observed I/O	Recover an unknown procedure
This study	Trace mean/std only	No plaintext, key, POI, structure, alignment marker, or code	Repetition scale, phase, stable core, segments

selected. Our work targets this earlier localization stage rather than the subsequent key-candidate scoring model.

Unlabeled POI search gives another point of comparison. Variance analysis labeling [3] and leakage-detection statistics [12] identify sample positions likely to carry data-dependent leakage. Such methods are valuable but operate mainly at the sample-POI level. Here the first output is a repetition-level coordinate system: period, phase, stable core, and relative segments. Alignment and reverse-engineering work is also adjacent. SCATTER [5] reduces sensitivity to jitter/shuffling once the target algorithm, intermediate value, key candidate, and leakage window have been set; Generic SCARE [6] reconstructs an unknown procedure from input, output, and execution traces. Generic SCARE is therefore closer to procedure reconstruction, whereas this work produces a segmentation and coordinate system for later side-channel analysis. In contrast, the proposed method uses only mean/std trace statistics and produces a compact candidate structure before any plaintext, ciphertext, key candidate, source/assembly reference, or execution marker is used.

Table 1 summarizes the position of this work relative to two nearby methods. These methods are not direct baselines because their input assumptions and output objectives differ; they are included to clarify the position of the proposed upstream structuring task.

3 Proposed Method

3.1 Overview

The method receives only the per-sample mean μ_t and standard deviation σ_t of measured traces,

$$\mathcal{D} = \{\, (t, \mu_t, \sigma_t) \mid t = 0, \ldots, N-1 \,\}. \tag{1}$$

It does not use the algorithm name, number of rounds, source code, assembly, external markers, plaintext/ciphertext, key, or ground-truth POIs. Stage 1 structures the full trace into repetition windows and returns a dominant repetition scale, start phase, anchor, and stable core. Stage 2 stacks stable core windows into a representative repetition and divides its relative time axis into high- and low-score segments. Figure 1 summarizes the flow.

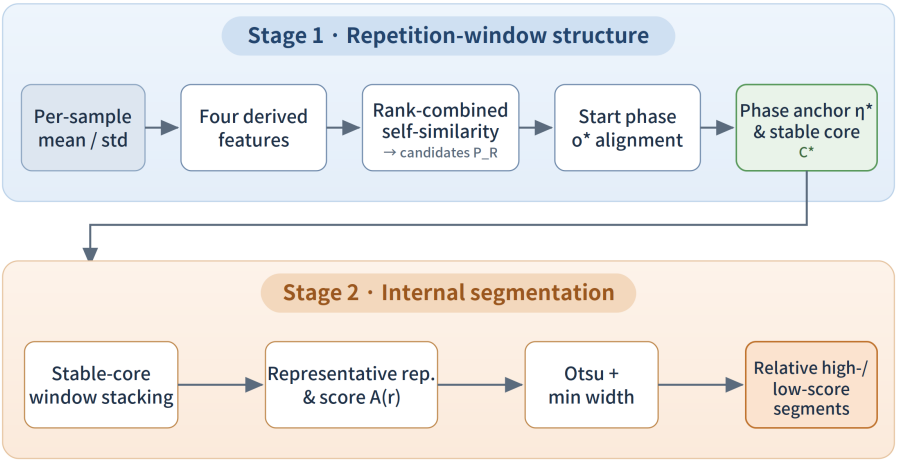


Fig. 1. The proposed two-stage trace-only repetition structuring process.

Table 2. Trace-only outputs and their intended interpretation.

Output	Role in the coordinate system	Use in later analysis
$\mathcal{P}_R$	Candidate repetition scales ranked by consensus self-similarity	Keeps alternative hierarchy levels when Rank-1 is a microperiod or grouped macro body
P^*, o^*	Rank-1 period and aligned start phase	Converts absolute samples into repetition index and relative position
$\eta^*, \mathcal{C}^*$	Recurring anchor and longest supported window run	Selects the windows used to form a stable representative repetition
$\{S_q\}$	Relative high-/low-score internal segments	Restricts the first regions examined by later CPA or BSCA without claiming ground-truth POIs

The design assumes that repeated execution leaves reproducible structure in the mean or standard deviation, that correctly cut repetition windows show high waveform consistency and recurring anchors, and that internal features are stable in relative rather than absolute time. The selected period is therefore an implementation-level repetition hierarchy visible in the trace statistics, not necessarily the exact semantic round.

3.2 Stage 1: Repetition-Window Structure Estimation

Four non-negative features capture waveform magnitude and boundary changes:

$$x_t^{(1)} = \sigma_t, \quad x_t^{(2)} = |\mu_t|, \quad x_t^{(3)} = |\nabla \mu_t|, \quad x_t^{(4)} = |\nabla \sigma_t|. \tag{2}$$

The channels combine magnitude and boundary views. Each feature is centered and normalized before autocorrelation; raw autocorrelation scores are converted

to within-channel percentile ranks, with larger ranks better, and then averaged:

$$C(\ell) = \frac{1}{4}\sum_{j=1}^{4} \mathrm{rank}_{\ell\in\mathcal{L}}\, R_j(\ell), \quad \mathcal{P}_R = \underset{\ell\in\mathcal{L}}{\text{TopR-LocalMax}}\, C(\ell), \quad P^* = P_1. \tag{3}$$

For candidate origin o, the k-th window starts at $o + kP^*$ and is represented by an in-window centered, L_2-normalized vector $z_{k,o}^{(j)}$. The alignment score is

$$B(o) = \frac{1}{4}\sum_{j=1}^{4} \underset{k<k'}{\mathrm{median}}\, \mathrm{corr}\left(z_{k,o}^{(j)}, z_{k',o}^{(j)}\right). \tag{4}$$

To ensure that windows share a recurring internal event, the median template $T_j(r)$ is searched for peak anchors $\eta = (j, a)$. The stable core is the longest consecutive run of anchor-supporting windows:

$$T_j(r) = \underset{k}{\mathrm{median}}\, x_{o^*+kP^*+r}^{(j)}, \quad \mathcal{C}^* = \mathrm{LongestConsecutiveRun}(\mathcal{K}_A(\eta^*)). \tag{5}$$

Thus Stage 1 outputs $(\mathcal{P}_R, P^*, o^*, \eta^*, \mathcal{C}^*)$. The stable core indicates continuous anchor support in the trace statistics; it is not an independently verified execution interval.

Procedure 1: Trace-only repetition estimation. Compute the four channels in Eq. 2. For every lag in $\mathcal{L}$, compute normalized autocorrelation per channel, convert each channel's lag scores to percentile ranks with larger ranks better, and average the ranks to obtain $C(\ell)$. A lag is a local maximum if its score is no smaller than both neighboring scores; the Top-R local maxima, ordered by $C(\ell)$, form $\mathcal{P}_R$, and $P^* = P_1$. Then scan 96 coarse origins, refine near the best origin, and among origins within the fixed correlation tolerance select o^* lexicographically by $(K(o), B(o), -o)$. Finally, enumerate median-template peaks, count a window as supporting an anchor only when a salient per-window peak falls within the fixed phase band, and choose η^* lexicographically by maximum support, minimum duplicate in-band peaks, minimum median phase error, channel, and phase. The longest consecutive supporting run is $\mathcal{C}^*$. All fixed thresholds are listed in Table 3.

3.3 Stage 2: Internal Segmentation of the Representative Repetition

Stage 2 keeps P^*, o^*, and $\mathcal{C}^*$ fixed. It stacks stable-core windows onto a common relative axis and builds a representative template. For relative position r,

$$\hat{T}_j(r) = \underset{k\in\mathcal{C}^*}{\mathrm{median}}\, x_{o^*+kP^*+r}^{(j)}, \quad A(r) = \frac{1}{4}\sum_{j=1}^{4} \mathrm{rank}_{u\in[0,P^*)}\left(\hat{T}_j(u)\right)\Big|_{u=r}. \tag{6}$$

$A(r)$ is a dimensionless prominence score, not a leakage-strength estimate for a specific intermediate value.

Procedure 2: Internal segmentation. Choose the longest anchor-supported run, stack its aligned windows, and compute $A(r)$ by averaging per-channel percentile ranks on the relative axis. Smooth $A(r)$, apply Otsu thresholding to obtain high/low labels,

Table 3. Fixed parameters used for all targets.

Stage	Fixed setting
Input	1,000 traces per target, summarized as mean/std from coarse invocation-trigger alignment only; no internal marker, round boundary, or PC trace
Period search	channels σ, $\lvert\mu\rvert$, $\lvert\nabla\mu\rvert$, $\lvert\nabla\sigma\rvert$; $\ell = 80, \ldots, \min(20000, \lfloor N/2 \rfloor - 1)$; percentile-rank autocorrelation consensus; local maxima; Top-$R = 5$
Phase/anchor	96 coarse origin steps plus local refinement; origin correlation tolerance 0.01; anchor peak quantile 0.97, spacing $0.01P^*$, phase band $\delta = 0.025P^*$
Internal segments	longest anchor-supported run; smoothing $0.0025P^*$; Otsu 128 bins; minimum width $0.002P^*$; short-run neighbor merge and window projection
Post-hoc use	plaintext/ciphertext, key, algorithm label, source/assembly, and POI labels are excluded during generation; source/ASM/CPA are used only after outputs are fixed

and repeatedly merge runs shorter than the fixed minimum width into a neighbor: edge runs use the only neighbor, equal-label neighbors are joined, otherwise the wider neighbor is chosen, with ties broken by closer mean score. The resulting activity/context intervals are projected to each selected window at the same relative positions.

The output segments are operational candidate regions:

$$\begin{aligned} &\mathcal{A} = \{\, r \mid \bar{A}(r) \geq \tau_{\text{Otsu}} \,\}, \qquad \{S_q\}_{q=1}^{Q} = \text{Segments}_{w_{\min}}(\mathcal{A}), \\ &\Theta = (\mathcal{P}_R,\, P^*,\, o^*,\, \eta^*,\, \mathcal{C}^*,\, \{S_q\}_{q=1}^{Q}). \end{aligned} \tag{7}$$

Inter-repetition support and boundary strength are reported only as descriptive indicators of the generated segments, not as additional generation criteria.

4 Evaluation

4.1 Experimental Setup and Metrics

We evaluate 16 targets: AES, GIFT, HIGHT, PRESENT, RECTANGLE, SIMON, SM4, and SPECK on STM32F303 and XMEGA. We assume that traces are coarsely capture-aligned to the target invocation, but no internal execution marker, round boundary, PC trace, plaintext/ciphertext, key, source code, assembly, or POI label is used by the method. For each target, multiple measured traces were summarized into per-sample mean and standard deviation before analysis. Table 3 summarizes the fixed trace-only conditions.

Stage 1 is measured by the number of aligned complete windows K, anchor-supporting windows K_A, stable-core length L_C, and waveform consistency B. For post-hoc source/assembly comparison, N_{ref} denotes the identical repeated body count in the compiled implementation. For each binary, N_{ref} was assigned post-hoc as the repeated source/ASM body implementing the main encryption-state update; when source loops and assembly repetition differed, the assembly-level body was used. Setup, finalization, whitening, and helper code were excluded unless structurally repeated in the same compiled loop. Relation labels in Table 4 were assigned only after the trace-only output was fixed. Target-level N_{ref} bases

are retained in the post-hoc source/assembly review records; this paper reports the compact count relation. A count c is direct if $|c - N_{\mathrm{ref}}| \leq 1$ and integer-ratio consistent if it is within one of mN_{ref} or N_{ref}/m for $m \in \{2, 3, 4, 5, 6, 8, 10, 16, 32\}$. These ratio checks are used only as post-hoc hierarchy annotations, not as semantic round-recovery success. Rank-1 uses $c \in \{K, L_C\}$; Top-5 uses the counts implied by candidate periods. These metrics measure structure, not key recovery or sample-level POI accuracy: K, K_A, L_C, and B describe internal waveform consistency and anchor continuity, while comparison with N_{ref} is a post-hoc count-level check against source/assembly. Rank-1 therefore reports the consistency of the single selected period, whereas Top-5 reports whether the candidate set still contains the implementation hierarchy when Rank-1 selects another visible layer.

4.2 Cross-Implementation Readout

Stage 1 first checks whether unlabeled traces can be converted into coherent repetition windows and a continuous stable core.

Figure 2 is used as a global stability overview for all targets; detailed waveform interpretation is reserved for the representative AES/XMEGA readout in Figure 3. In 12/16 cases, $K = K_A = L_C$, so every aligned window belonged to one continuous stable core. The median B was 0.989, and 12 targets had $B \geq 0.97$. Thus Rank-1 usually produced both waveform similarity and anchor continuity, while a few targets exposed either low consistency or a short core.

Figure 3 shows AES/XMEGA, where $P^* = 4{,}400$, $o^* = 572$, and all 9 aligned windows were included in the stable core. The waveform consistency was 0.994 and the median correlation to the standard-deviation template was 0.996. This detailed exact-case readout complements the global overview in Figure 2. PRESENT/F303, RECTANGLE/XMEGA, and SM4/XMEGA instead produced short stable cores, while GIFT/XMEGA retained all windows but with low waveform consistency. This separation shows why both waveform consistency and anchor continuity are needed.

4.3 Post-hoc Source/Assembly Comparison

Table 4 compares the fixed Rank-1 trace-only counts with the post-hoc source and assembly body count. It separates exact and edge-inclusive cases from grouped, microperiod, and hierarchy-ambiguous outcomes. Thus Rank-1 is reported as structural or hierarchy readout, not as semantic round recovery.

The table contains five exact body-count matches and two edge-inclusive cases, where setup, whitening, or boundary code appears as body-like windows. Other targets reflect implementation hierarchy: HIGHT, SIMON, and SM4 show grouped macro relations, while PRESENT/F303, RECTANGLE/XMEGA, and SM4/XMEGA are dominated by short instruction- or bit-level motifs. In 14/16 targets, a Top-5 candidate is body-count-related; this is a post-hoc hierarchy check, not a claim that source knowledge selects the candidate.

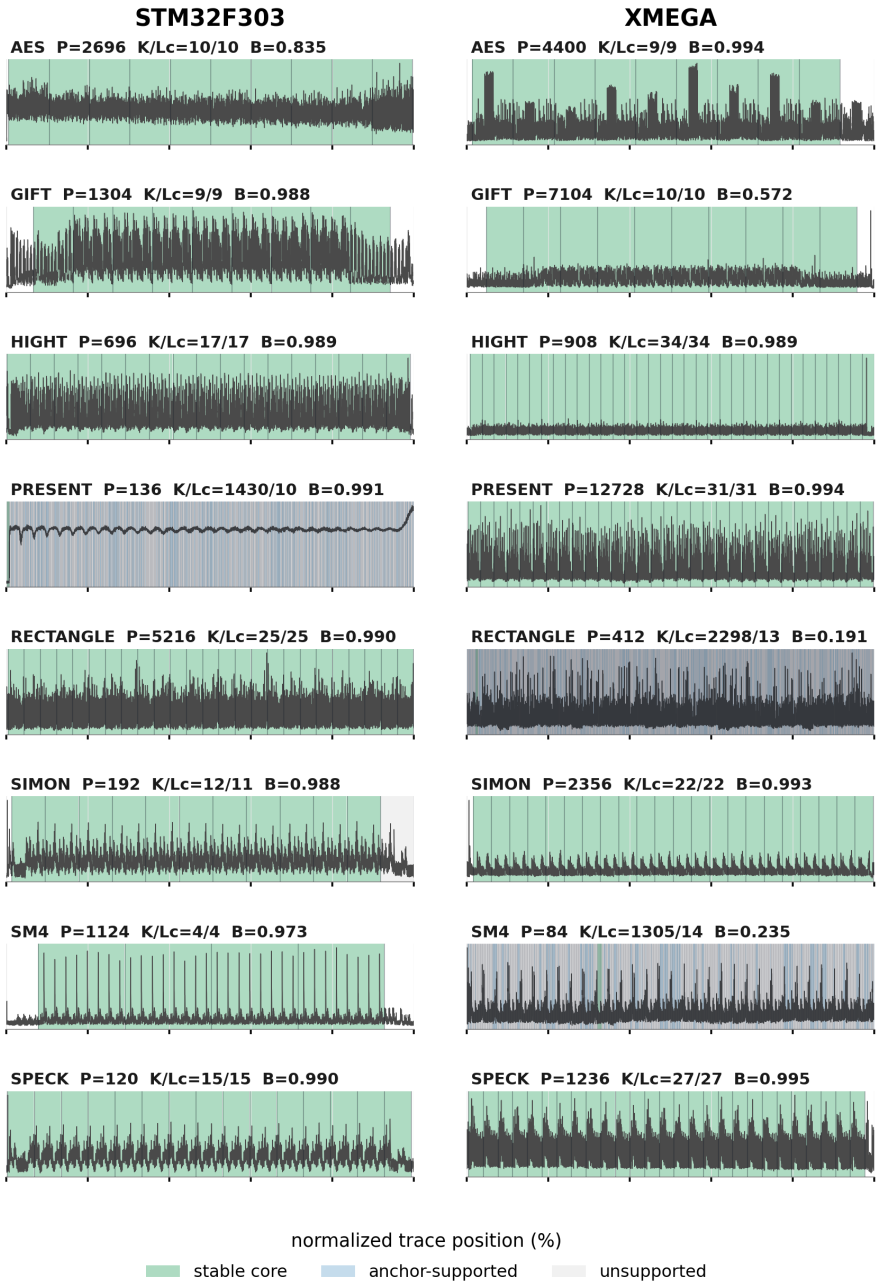


Fig. 2. Rank-1 trace-only repetition-window segmentation results for 16 targets. Green denotes the stable core, blue the anchor-supporting windows outside the stable core, and gray the anchor-unsupported aligned windows. The figure is intended to show global stability rather than detailed waveform differences.

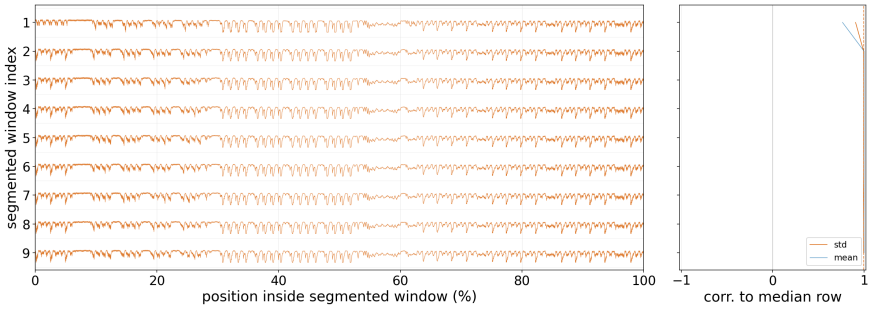


Fig. 3. Representative exact/stable case: standard-deviation waveforms of the 9 aligned AES/XMEGA repetition windows (left) and their correlation to the median repetition window (right).

Table 4. Post-hoc Rank-1 count relation to source/assembly repeated bodies, grouped by platform.

Cipher	K/L_C	Ref.	Relation	Main interpretation
STM32F303				
AES	10/10	9	edge-incl.	edge-inclusive body
GIFT	9/9	7	hier. ambiguous	macro/helper cadence
HIGHT	17/17	32	grouped hierarchy	two stages per window
PRESENT	1430/10	31	microperiod	short S-box/permutation motif
RECTANGLE	25/25	25	exact	body count matched
SIMON	12/11	22	grouped hierarchy	two loop bodies per window
SM4	4/4	8	grouped hierarchy	two 4-round macro blocks
SPECK	15/15	27	hier. ambiguous	grouped ARX cadence
XMEGA				
AES	9/9	9	exact	body count matched
GIFT	10/10	7	hier. ambiguous	loose macro cadence
HIGHT	34/34	32	edge-incl.	edge/whitening included
PRESENT	31/31	31	exact	body count matched
RECTANGLE	2298/13	25	microperiod	bit/permutation motif
SIMON	22/22	22	exact	body count matched
SM4	1305/14	8	microperiod	macro body in Top-5
SPECK	27/27	27	exact	body count matched

Ref. follows the N_{ref} assignment rule in Section 4; it is not necessarily the semantic round count. `hier. ambiguous` means that the Rank-1 count alone does not identify the source/ASM body count.

4.4 Internal Segmentation Case Study

For AES/XMEGA, known-key CPA is performed only after the trace-only internal segments are fixed, and is used only for post-hoc interpretation. The CPA models are HW(PT), HW($PT \oplus K$), and HW(Sbox($PT \oplus K$)). At each relative sample we use the maximum absolute correlation over 16 bytes. Hotspot coverage is the fraction of top-p CPA samples falling in the fixed high-score set $\mathcal{A}$.

Figure 4 shows the relative activity profile and the internal segments of the representative repetition formed from 9 aligned windows. The same relative seg-

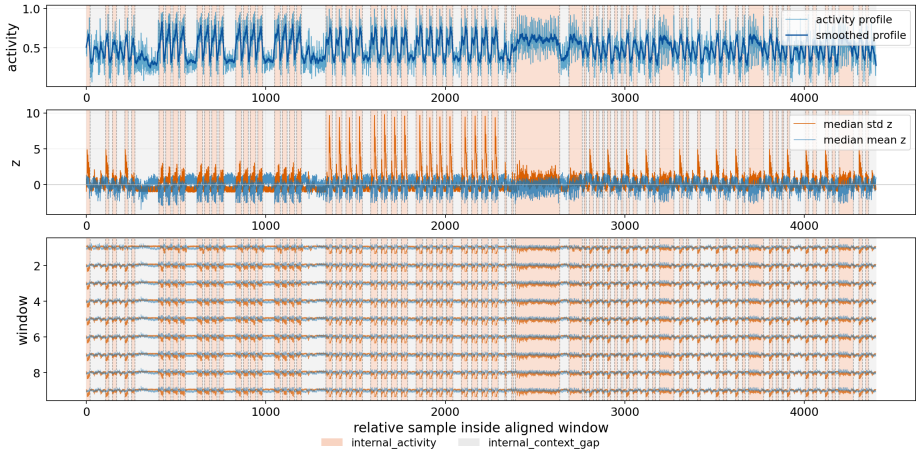


Fig. 4. Internal segmentation of the representative repetition: the relative activity profile and its smoothed version (top), the median standardized mean and standard-deviation features (middle), and the resulting internal segments commonly applied to the aligned repetition windows (bottom).

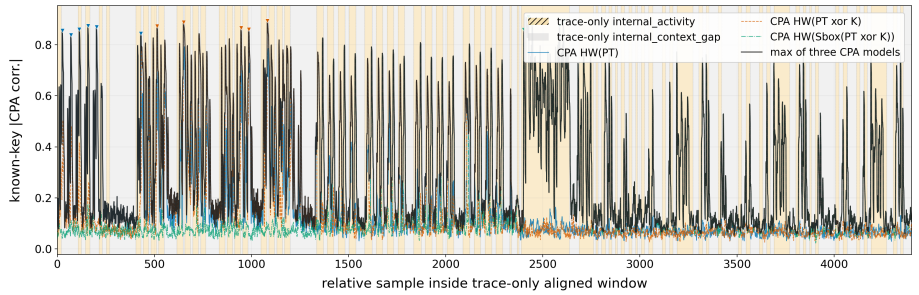


Fig. 5. Correspondence between the fixed internal segments in body 01 and the HW(PT), HW($PT \oplus K$), and HW(Sbox($PT \oplus K$)) CPA waveforms.

ments are projected to the median features and to each individual window, confirming a shared internal coordinate system.

Figure 5 shows the result of post-hoc overlaying CPA waveforms on the fixed body-01 segments. The compact coverage summary below reports the same result numerically; high-score segments occupy 49.1% of the representative body.

CPA model	**Top 0.5%**	**Top 0.1%**
HW(PT)	45.5%	60.0%
HW($PT \oplus K$)	0.0%	0.0%
HW(Sbox($PT \oplus K$))	100.0%	100.0%

The Stage 2 segments therefore capture a strong S-box CPA hotspot while remaining candidate regions rather than ground-truth POIs. This representative case study does not establish cross-target CPA reduction; it tests whether fixed trace-only segments can align with a known leakage hotspot in one target.

4.5 Discussion and Limitations

The experiments show that Rank-1 and Top-5 results should be interpreted at different levels. Rank-1 evaluates the single period that the unsupervised procedure would use by default. It is therefore the strictest trace-only output: if it matches the source/assembly body count, the method has found a directly usable implementation-level cadence. Top-5, in contrast, is not selected using source-level knowledge. It is an ordered set of alternative trace-only coordinate systems; downstream analysis can inspect candidates in that order and deprioritize candidates with very short cores, extreme window counts, or low waveform consistency. This distinction matters because embedded implementations often contain several valid repetition scales at once: loop bodies, unrolled macro bodies, table-lookup motifs, permutation substeps, and short instruction sequences can all recur periodically in the power trace.

The main failure modes are also informative. An edge-inclusive count, as in AES/STM32F303 or HIGHT/XMEGA, indicates that setup or whitening-like code has a waveform similar enough to the body to be absorbed into the window sequence. A grouped count indicates that the visible cadence combines several semantic rounds or loop bodies into one trace window. A microperiod indicates that an inner operation dominates the mean/std statistics more strongly than the algorithmic body. A short stable core means that only a contiguous subset of the aligned windows maintains the selected anchor. These cases are not equivalent to sample-level segmentation errors; rather, they identify which implementation layer is most visible under the trace-only statistics.

For downstream CPA or BSCA, the intended use is conservative: try Rank-1 first, inspect Top-5 alternatives in trace-only order when Rank-1 is short, unstable, or too numerous, and use $\{S_q\}$ only to prioritize the first search region. The AES/XMEGA overlay covers one strong S-box hotspot, but other key-dependent correlations may still lie outside the high-score set; the output is therefore a prioritization mechanism, not a final POI oracle.

The method provides a trace-only candidate coordinate system, not a complete key recovery attack. P^* may select a round, macro body, harmonic period, or instruction-level microperiod; o^* is not sample-level verified without an execution marker or PC trace; and $\{S_q\}$ denotes relative statistical prominence, not exhaustive leakage locations. The next step is to connect this coordinate system to blind labeling and key-inference procedures and measure the resulting search-cost reduction.

5 Conclusion

This paper defined the trace-only structuring problem that precedes BSCA and proposed a two-stage trace-only method using only per-sample mean and standard deviation. The method estimates a repetition scale, start phase, anchor-supported stable core, and relative high-/low-score segments, thereby transforming an unlabeled trace into a hierarchical repetition coordinate system. Across

16 block-cipher implementations, 12 targets had continuous stable cores and 12 reached high waveform consistency. Post-hoc source/assembly comparison separated exact and edge-inclusive count matches from grouped, microperiod-dominant, and hierarchy-ambiguous Rank-1 relations, while Top-5 candidates retained a body-count-related hierarchy in 14 targets. In one representative AES/XMEGA case, the fixed high-score segments also covered the strongest S-box-based CPA hotspots. These results support the method as a preliminary trace-only prioritization stage; future work will connect the produced coordinates to full blind key-inference pipelines and measure cross-target search-cost reduction.

References

1. Linge, Y., Dumas, C., Lambert-Lacroix, S.: Using the joint distributions of a cryptographic function in side channel analysis. In: Constructive Side-Channel Analysis and Secure Design (COSADE 2014). LNCS, vol. 8622, pp. 199–213. Springer (2014). https://doi.org/10.1007/978-3-319-10175-0_14
2. Le Bouder, H., Lashermes, R., Linge, Y., Thomas, G., Zie, J.-Y.: A multi-round side channel attack on AES using belief propagation. In: Foundations and Practice of Security (FPS 2016). LNCS, vol. 10128, pp. 199–213. Springer (2017). https://doi.org/10.1007/978-3-319-51966-1_13
3. Clavier, C., Reynaud, L.: Improved blind side-channel analysis by exploitation of joint distributions of leakages. In: Cryptographic Hardware and Embedded Systems (CHES 2017). LNCS, vol. 10529, pp. 24–44. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_2
4. Clavier, C., Reynaud, L., Wurcker, A.: Quadrivariate improved blind side-channel analysis on Boolean masked AES. In: Constructive Side-Channel Analysis and Secure Design (COSADE 2018). LNCS, vol. 10815, pp. 153–167. Springer (2018). https://doi.org/10.1007/978-3-319-89641-0_9
5. Thiebeauld, H., Gagnerot, G., Wurcker, A., Clavier, C.: SCATTER: a new dimension in side-channel. In: Constructive Side-Channel Analysis and Secure Design (COSADE 2018). LNCS, vol. 10815, pp. 135–152. Springer (2018). https://doi.org/10.1007/978-3-319-89641-0_8
6. Lashermes, R., Le Bouder, H.: Generic SCARE: reverse engineering without knowing the algorithm nor the machine. Journal of Cryptographic Engineering **14**(2), 399–414 (2024). https://doi.org/10.1007/s13389-024-00356-2
7. Azouaoui, M., Papagiannopoulos, K., Zürner, D.: Blind side-channel SIFA. In: 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 555–560. IEEE (2021). https://doi.org/10.23919/DATE51398.2021.9474245
8. Sarry, M., Le Bouder, H., Maaloouf, E., Thomas, G.: Blind side channel analysis against AEAD with a belief propagation approach. In: Smart Card Research and Advanced Applications (CARDIS 2023). LNCS, vol. 14530, pp. 127–147. Springer (2024). https://doi.org/10.1007/978-3-031-54409-5_7
9. Ravi, P., Jap, D., Bhasin, S., Chattopadhyay, A.: Invited paper: machine learning based blind side-channel attacks on PQC-based KEMs — a case study of Kyber KEM. In: 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD), pp. 1–7. IEEE (2023). https://doi.org/10.1109/ICCAD57390.2023.10323721

10. Rezaeezade, A., Yap, T., Jap, D., Bhasin, S., Picek, S.: Breaking the blindfold: deep learning-based blind side-channel analysis. In: 34th USENIX Security Symposium (USENIX Security 25), pp. 5777–5796. USENIX Association (2025)
11. Lee, I., Kim, G., Hong, S., Kim, H.: MALeak: blind side-channel key recovery exploiting modular addition leakage in ARX-based block ciphers. IACR Cryptology ePrint Archive, Paper 2026/067 (2026)
12. Durvaux, F., Standaert, F.-X.: From improved leakage detection to the detection of points of interests in leakage traces. In: Advances in Cryptology — EUROCRYPT 2016. LNCS, vol. 9665, pp. 240–262. Springer (2016). https://doi.org/10.1007/978-3-662-49890-3_10