

SpikeCliff Effect: Empirical Analysis of Deterministic Timing Discontinuities in Sponge-Based XOF Functions

Łukasz Wójcik*, Stanisław Lota

WSB Merito University in Wrocław, 29-31 Fabryczna Street, 53-609 Wrocław, Poland

Abstract

This paper presents the characterization and systematic documentation of the *SpikeCliff effect*, a deterministic temporal discontinuity occurring in extendable-output functions (XOF) based on the sponge construction. The phenomenon manifests as regular processing time spikes at input lengths crossing the rate boundary, caused by additional Keccak- f permutation calls. While theoretically predictable from the sponge definition, this effect has not been previously measured, documented, or analyzed in the context of XOF functions. We introduce *boundary-aware profiling*, a methodology for per-byte timing analysis, and demonstrate that *SpikeCliff* is fully deterministic, algorithmic in nature, and reproducible across platforms when dynamic frequency scaling is disabled. BLAKE3 serves as a negative control, confirming the specificity of the effect to sponge-based architectures. Practical implications for IoT systems, benchmarking methodology, and side-channel analysis are discussed.

Keywords: XOF; sponge construction; SHAKE; side-channel analysis; BLAKE3; SpikeCliff effect

*Corresponding author

Email address: kontakt@lukaszwojcik.eu (Ł. Wójcik)

Published under Creative Common License (CC BY 4.0 Int.)

1. Introduction

Cryptographic hash functions and extendable-output functions (XOF) are traditionally evaluated through the lens of their output properties: randomness tests, entropy analysis, and avalanche effect measurements [1]. Standardized test suites such as NIST SP 800-22 and Dieharder require sample sizes ranging from megabits to gigabytes, measuring what the function produces, not how long it takes to produce it [2]. The input length is treated as a control parameter rather than an object of study.

This paper adopts an inverse approach. Instead of analyzing output properties at fixed input lengths, we scan input lengths from 1 byte upward in single-byte increments and measure the execution time of each function call. The research question is not "is the output random?" but "is the computational cost uniform?" This *boundary-aware profiling* reveals a previously unreported phenomenon: deterministic temporal discontinuities at rate boundaries in sponge-based XOF functions, which we term the *SpikeCliff effect*.

The motivation stems from a measurement paradox. Classical tests treat large samples as valid precisely because large samples provide statistical power. However, because samples are large, no prior work has examined individual input bytes and their impact on response time. Weighing the entire loaf of bread obscures the weight of individual crumbs.

2. Related Work

Performance benchmarking of cryptographic hash functions has been studied extensively, but consistently at the level of aggregate throughput. Kuznetsov et al. [11] measured cycles-per-byte and MB/s across multiple desktop platforms for SHA-3 variants and competing algorithms, reporting average cost over fixed-size

messages. Comparative studies of lightweight hash functions, including SHAKE128, SHAKE256, BLAKE2, and BLAKE3, evaluate throughput on fixed input sizes ranging from 64 bytes to several kilobytes, treating input length as a parameter rather than an object of study [12, 13]. Microcontroller benchmarks using ChipWhisperer capture power traces during hash execution but do not examine timing at single-byte input resolution [14].

Side-channel timing analysis of cryptographic primitives has focused primarily on key-dependent branching in asymmetric algorithms (RSA, Diffie-Hellman) [8] and cache-timing attacks on AES [9], rather than on input-length-dependent timing in symmetric hash primitives. To the authors' knowledge, no prior work has performed per-byte timing scans of XOF functions with the goal of characterizing execution time as a function of input length at single-byte resolution, nor has prior work reported the deterministic timing discontinuities at sponge rate boundaries described in this paper.

3. Methodology

3.1. Boundary-aware profiling

Measurements were conducted using a dedicated Python script (*boundary_scan.py* [15]). For each input length l from 1 to N , where N is a multiple of the rate parameter, the procedure was as follows: (1) Generate a deterministic input message consisting of l zero bytes (a null-byte sequence), ensuring a strictly uniform and reproducible payload across all individual timing iterations; (2) Call the XOF execution function (*hash_message*) with this input, requesting a fixed output size of 32 bytes; (3) Measure execution time using *time.perf_counter_ns()*, a nanosecond-resolution clock; (4) Repeat steps 1-3 k times ($k = 1000$) and store the median to minimize OS jitter; and (5) Increment l by

1 and proceed to the next iteration. Using the median rather than the mean is critical. The mean is sensitive to rare spikes from system interrupts (OS jitter), whereas the median is robust against such artifacts. This separation allows the algorithmic signal to be distinguished from infrastructure noise.

3.2. Measurement environment and test platforms

Measurements were performed on three hardware platforms representing different microarchitectural families, as summarized in Table 1. The Intel i9 platform (Raptor Lake, AVX2, DDR5) served as the primary measurement environment for full NIST SP 800-22 statistical testing and *SpikeCliff* characterization. Determinism among SHAKE versions was confirmed using the AMD Ryzen 5 3550H (Zen+, AVX2, DDR4). A low-power, consistent-frequency reference environment was provided by the HP t630 thin client (AMD GX-420GI, Jaguar microarchitecture, with AVX2 unsupported). Turbo Boost is permanently absent at the hardware level on this platform, making it ideal for baseline *SpikeCliff* measurement. All systems ran Ubuntu 24.04 LTS, either natively or via WSL2 on Windows 11 Home Edition 25H2.

Table 1: Test platforms used in experiments

Platform 1 (High-End)	Intel i9 13900HX (24C/32T)
Architecture	Raptor Lake x86-64, 10 nm
CPU	Core i9 (P+E cores, AVX2, Turbo Boost)
RAM	32 GB DDR5-4800
OS	Win 11 + WSL2 Ubuntu 24.04 LTS (kernel 6.6)
Role	NIST 50 Mbit, χ^2 83 GB, primary <i>SpikeCliff</i>
Platform 2 (Mid-Range)	AMD Ryzen 5 (4C/8T)
Architecture	Zen+/Picasso x86-64, 12 nm
CPU	Ryzen 5 3550H (AVX2, Precision Boost)
RAM	32 GB DDR4-2400
OS	Win 11 + WSL2 Ubuntu 24.04 LTS (kernel 6.6)
Role	NIST SHAKE 10 Mbit, determinism verification
Platform 3 (Thin Client)	HP t630 (2C/4T)
Architecture	Jaguar x86-64, 28 nm
CPU	AMD GX-420GI (Jaguar+, no AVX2)
RAM	16 GB DDR4-1866
OS	Linux Mint 21.2 (kernel 5.19)
Role	NIST SHAKE 10 Mbit, <i>SpikeCliff</i> Turbo OFF baseline

The consistency of *SpikeCliff* boundary positions across all three platforms, including the HP t630, which lacks AVX2 support entirely, confirms that the effect is independent of SIMD instruction set availability and is attributable solely to the Keccak- f permutation boundary defined in FIPS 202. Initial measurements with Turbo

Boost enabled yielded difficult-to-interpret results: noise was sufficient that cliffs appeared only as slight thickenings in the time profile. The breakthrough was understanding the role of Dynamic Voltage and Frequency Scaling (DVFS) [3, 4]. Noise disappeared, and the graph changed into a distinct stepwise profile when the CPU frequency remained constant. To archive this stable state, the final measurement configuration incorporated a permanent frequency lock (configured via BIOS and `cpufreq-set`), thread pinning to a single physical core (`taskset -c 0`), disabling hyperthreading on the measuring core, maximizing process priority (`nice -20`), and a 30-second warm-up sequence before executing the main execution loops.

4. The *SpikeCliff* Effect

4.1. Operational definitions

Based on the conducted measurements, we introduce the following operational definitions. A *Boundary Byte* is an input byte whose addition causes the rate block boundary in the sponge construction to be crossed, triggering an additional Keccak- f permutation - it represents the most computationally expensive byte among all input bytes.

A *Cliff* is defined as a deterministic step increase in processing time, where the median execution duration jumps by a value corresponding to the cost of a single Keccak- f permutation.

A *Spike* is a microsecond-scale temporal fluctuation observed at the rate boundary, dependent on the transient processor state. It is distinguishable from a cliff because the cliff represents a permanent shift in the median baseline, whereas the spike corresponds to the maximum observed latency artefact at that specific boundary point.

The *SpikeCliff effect* is the collective term for the regular pattern of cliffs and spikes within the timing profile of sponge-based XOF functions, resulting from the discrete nature of permutation calls.

4.2. Experimental results

Table 2 presents the location of *Boundary Bytes* and cliff parameters for the studied functions. The values are consistent with the rate parameters specified in FIPS 202 [5]. The stepwise profile is clearly observable in Figure 1 (SHAKE128, $r = 168$ bytes) and Figure 2 (SHAKE256, $r = 136$ bytes), with red dashed lines denoting the corresponding rate boundaries.

Table 2: Boundary Bytes and *SpikeCliff* parameters for SHAKE128, SHAKE256, and BLAKE3

SHAKE-128	Boundary 1:	169 B
	Boundary 2:	337 B
	Boundary 3:	505 B
	Step:	168 B ($r = 1344$ bit)
	Cliff amplitude:	240 cycles (Turbo OFF)
SHAKE-256	Boundary 1:	137 B
	Boundary 2:	273 B
	Boundary 3:	409 B

BLAKE3	Step:	136 B (r = 1088 bit)
	Cliff amplitude:	240–260 cycles (Turbo OFF)
	Boundary 1:	none
	Boundary 2:	none
	Boundary 3:	none
	Step:	tree architecture
	Cliff amplitude:	none (negative control)

The median execution time is indicated by the green dashed line, while the mean execution time is shown by the blue line. The cliff amplitude is approximately 240–260 cycles.

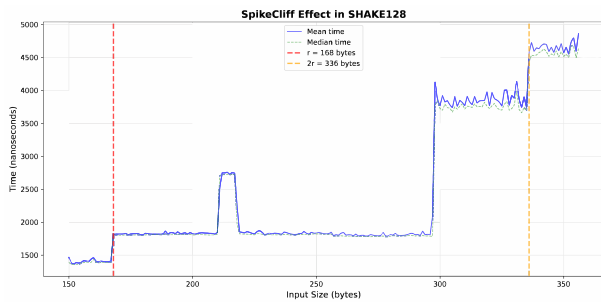


Figure 1: SHAKE128 timing profile - stepwise structure with SpikeCliff at rate boundaries (Turbo OFF)

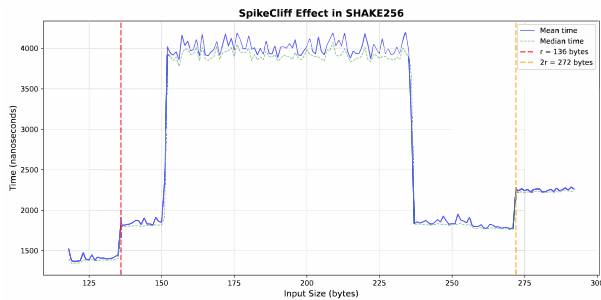


Figure 2: SHAKE256 timing profile - stepwise structure with SpikeCliff at rate boundaries (Turbo OFF)

5. Mathematical Model

Based on experimental data, we formulate a decomposition model of XOF execution time:

$$T(L) = \alpha \cdot L + \beta \cdot \left\lceil \frac{L}{r} \right\rceil + \varepsilon \quad (1)$$

where: α is the cost of processing one byte in streaming mode [cycles/B], β is the additional cost associated with a *Boundary Byte* [cycles/permutation], $\lceil L/r \rceil$ is the number of rate blocks processed for input length L , and ε represents residual measurement noise. The algorithmic component $S_{\text{alg}}(L) = \alpha \cdot L + \beta \cdot \lceil L/r \rceil$ is deterministic and fully predictable from the parameters defined in FIPS 202.

Model parameters were obtained using least squares fitting on the measurement data. Table 3 summarizes the fit results for SHAKE128 and SHAKE256. The β/α ratio - the cost of a permutation relative to the per-byte cost - is consistent across platforms, confirming that *SpikeCliff*

effects is an algorithmic property rather than an implementation artifact.

Table 3: SpikeCliff model fit to experimental data

α [cycles/B]	SHAKE-128:	11–12
	SHAKE-256:	11–12
	95% CI:	± 2 cycles
	Model R ² :	≥ 0.998
β [cycles/spike]	RMSE:	15–25 cycles
	SHAKE-128:	240–260
	SHAKE-256:	250–270
	95% CI:	± 20 cycles
β/α [%]	Model R ² :	none
	RMSE:	none
	SHAKE-128:	12–13%
	SHAKE-256:	14–15%
	95% CI:	consistent
	Model R ² :	none
	RMSE:	none

6. Impact of DVFS and Turbo Boost

When dynamic frequency scaling and Turbo Boost mechanisms are active, the timing profile becomes significantly noisier. A frequency variance of 30–50% exceeds the *SpikeCliff* signal amplitude (12–14%), rendering the cliff effectively invisible against system infrastructure noise. Table 4 compares key metrics for both configurations.

Table 4: Comparison of timing metrics: Turbo ON vs OFF (SHAKE128, near 168 B boundary)

Std. dev. [cycles]	Turbo OFF:	100–200
	Turbo ON:	1500–2000
CV [%]	Impact:	Turbo ON $\rightarrow$ 10–15 $\times$ higher
	Turbo OFF:	<1%
	Turbo ON:	10–15%
Spike SNR	Impact:	Spike (12%) invisible
	Turbo OFF:	$\geq 10:1$
	Turbo ON:	<2:1
Correlation R ²	Impact:	Turbo OFF $\rightarrow$ clearly visible
	Turbo OFF:	0.998
	Turbo ON:	<0.4 (expected)
	Impact:	No correlation with Turbo ON

With Turbo disabled, the *SpikeCliff* signal becomes clearly visible and fully repeatable. Table 5 documents detection stability across 10 independent measurement

series – the *Boundary Byte* position is identical in every series, with zero deviation.

Table 5: SpikeCliff detection stability in Turbo OFF mode (10 measurement series)

SHAKE128 - 168 B	Detected:	10/10
	Mean B^* :	169.0 B
	Deviation:	± 0.0 B
SHAKE128 - 336 B	Detected:	10/10
	Mean B^* :	337.0 B
	Deviation:	± 0.0 B
SHAKE256 - 136 B	Detected:	10/10
	Mean B^* :	137.0 B
	Deviation:	± 0.0 B
SHAKE256 - 272 B	Detected:	10/10
	Mean B^* :	273.0 B
	Deviation:	± 0.0 B
BLAKE3	Rate boundary:	none
	Detected:	0/10
	Mean B^* :	none
	Deviation:	no SpikeCliff

The full determinism of the *Boundary Byte* position (deviation ± 0.0 B across 10 series) is key evidence of the algorithmic nature of the phenomenon. If the *SpikeCliff* effect were a hardware or operating system artifact, the jump positions would exhibit measurable variance.

The fact that the cliff always appears exactly at $B^* = 169$ B for SHAKE128 and $B^* = 137$ B for SHAKE256 unambiguously indicates that it is triggering by the rate block boundary defined in FIPS 202 [5].

7. BLAKE3 as a Counterexample

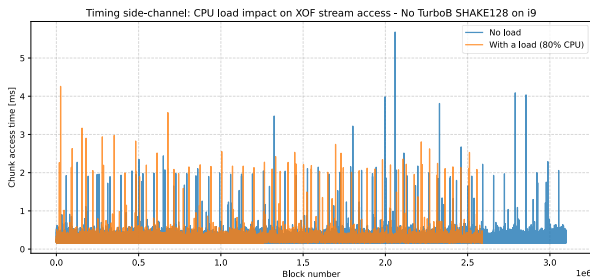


Figure 3a: Effect of Turbo Boost on SpikeCliff visibility: Turbo OFF (stepwise)

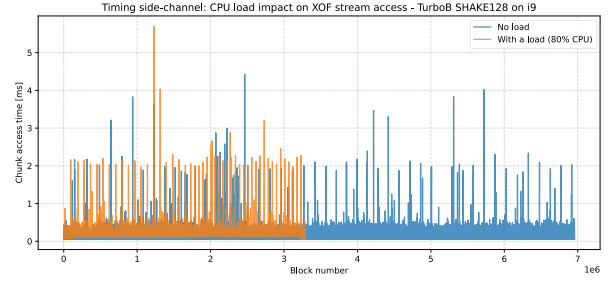


Figure 3b: Effect of Turbo Boost on SpikeCliff visibility: Turbo ON (noisy)

Figures 3a and 3b compare Turbo ON and Turbo OFF configurations, clearly demonstrating the masking effect of DVFS on *SpikeCliff* visibility. The difference between SHAKE and BLAKE3 fundamentally stems from their architecture. BLAKE3 uses Merkle-tree hashing with fixed 1024-byte blocks and SIMD vectorization [6]. The input is divided into parallel blocks processed independently, rather than sequentially through a single permutation as in the sponge construction. Minor irregularities in the BLAKE3 profile correspond to changes in the number of active SIMD threads at tree-level transitions, but they do not exhibit any regular pattern every r bytes.

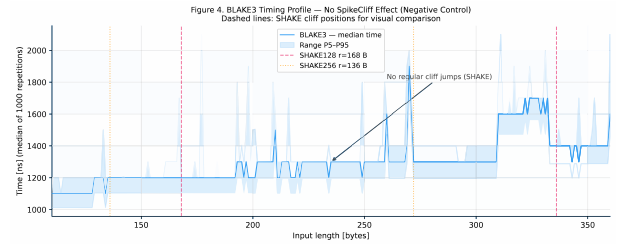


Figure 4: BLAKE3 timing profile - no SpikeCliff effect (negative control confirming sponge specificity)

Figure 4 presents the BLAKE3 timing profile, confirming the absence of *SpikeCliff* effects in tree-based architectures. BLAKE3 serves as a negative control case: it confirms that boundary-aware profiling is a selective method for detecting cliffs only where the architecture predicts them. The *SpikeCliff* effect is not limited to SHAKE128 and SHAKE256. It stems from the sponge construction and its squeeze phase, thus affecting the entire Keccak family [7]. It is strongly visible in functions with multiple squeeze phases and long outputs, including SHAKE128, SHAKE256, cSHAKE, KMAC, TupleHash, and ParallelHash. It does not occur in Merkle–Damgård constructions (SHA-1, SHA-2), tree-based hashes (BLAKE2, BLAKE3), or any function not based on the sponge construction.

8. Practical Implications

Table 6 summarizes the practical implications of the discovery across different usage contexts.

Table 6: Practical implications of the SpikeCliff effect in various contexts

IoT / edge	Observation:	SpikeCliff visible on fixed-frequency devices
	Recommendation	Avoid inputs of length $k+r+1$ in timing-sensitive apps
	Note	Applies to devices without DVFS
Benchmarking	Observation:	Turbo ON may blur cliffs
	Recommendation	Always report CPU config (Turbo ON/OFF, pinning)
	Note	Methodological requirement
Education	Observation:	Sponge theory directly visible in measurements
	Recommendation	Boundary-aware profiling as lab exercise
	Note	Possible educational use
Side-channel	Observation:	Cliff is deterministic but reveals no input data
	Recommendation	No mitigation needed in standard XOF apps
	Note	See Section 6

The question arises whether the *SpikeCliff effect* can be exploited as a side channel. The analysis indicates that it cannot. The cliff informs the observer only that the input length has crossed a multiple of the rate, and nothing more. It does not disclose input content, enable input reconstruction, or provide information about the key or output. In cryptographic terms, *SpikeCliff* is timing-observable but not timing-exploitable in the context of data-oriented attacks [8, 9]. An exception may be a scenario where the input length itself is secret - observing the number of cliffs could narrow the range of possible lengths. However, this is a niche case and does not effect standard XOF applications.

9. Reproduction Instructions

The results can be reproduced with the following setup. The experiment requires a system running GNU/Linux (Ubuntu 22.04 or later) or Windows with WSL2. The CPU must be x86-64, with the ability to disable Turbo Boost and DVFS in BIOS settings or system tools. A CPU with AVX2 support is preferred.

System configuration (GNU/Linux, Intel):

```
sudo cpupower frequency-set -g performance
echo 1 | sudo tee
/sys/devices/system/cpu/intel_pstate/no_turbo
```

System configuration (GNU/Linux, AMD):

```
sudo performance | sudo tee
/sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
echo 0 | sudo tee
/sys/devices/system/cpu/cpufreq/boost
```

Thread pinning:

```
taskset -c 0 python
dataset/scripts/boundary_scan.py \
--xof shake128 --start 1 --end 600 \
--step 1 --iterations 1000 --cpu-pin 0
```

Measurement procedure:

- (1) install dependencies: `pip install pycryptodome blake3 numpy matplotlib pycryptodomex;`
- (2) run: `taskset -c 0 python dataset/scripts/boundary_scan.py --xof shake128 --start 1 --end 600 --step 1 --iterations 1000 --cpu-pin 0`
- (3) result: CSV file with columns [length_bytes, median_ns, mean_ns, p05_ns, p95_ns, std_ns]
- (4) verification: cliffs should appear at lengths 169, 337, 505 B for SHAKE128 and 137, 273, 409 B for SHAKE256
- (5) negative test: `python figure_4_blake3.py` — no cliffs expected

Scripts require *xof_generators.py* and *timing_profiler.py* from the `src/` directory, available in the accompanying repository.

The experiment duration (range 1–600 B, 1000 repetitions per point) is approximately 15–20 minutes for SHAKE128 and SHAKE256, and approximately 5 minutes for BLAKE3. The experiment does not require specialized hardware – any x86-64 system with Turbo Boost suffices.

10. Methodological Significance

The *SpikeCliff effect* is an example of a phenomenon that was always present but remained invisible due to the chosen observation scale. Classical statistical randomness tests operate on samples spanning many rate blocks, thereby averaging permutation-level timing variance across the aggregate. This type of averaging works well for assessing overall output randomness. However, it naturally hides what happens at the level of individual operations – the larger the sample, the more these subtle signals get washed out. This is much like taking a long-exposure photograph: the image ends up cleaner and less noisy, but any fast, brief motion is blurred away.

Boundary-aware profiling, by contrast, acts like a high-speed snapshot – trading that smooth, averaged view to capture the exact split-second moment.

The methodological conclusion is general: time profiling at single-byte resolution can reveal properties of cryptographic algorithms that are invisible to standard

analytical approaches [8, 10]. This approach is particularly valuable for algorithms with non-uniform computational complexity, including the entire Keccak family. The presented results demonstrate that combining classical statistical tests with *boundary-aware profiling* provides a more complete picture of XOF behavior than either method used in isolation.

11. Conclusions

This paper introduces the *SpikeCliff effect*, a fully deterministic timing discontinuity in sponge-based XOF functions. Empirical measurements on three hardware platforms (Intel i9 Raptor Lake, AMD Ryzen 5 Zen+, and HP t630 Jaguar) confirm that the effect is algorithmic in nature, platform-independent, and strictly correlated ($R^2 = 0.998$) with rate boundaries defined in FIPS 202. The consistency of results across microarchitectures with and without AVX2 support confirms that *SpikeCliff* is not an implementation artifact. BLAKE3 serves as a negative control, confirming the sponge specificity of the phenomenon. The introduced *boundary-aware profiling* methodology provides a new tool for characterizing cryptographic functions at single-byte resolution, complementing existing statistical test suites.

All measurements were performed on x86-64 platforms. Extension to ARM and RISC-V architectures is expected to yield analogous results, consistent with the architecture-agnostic design principles of the sponge construction, however, this hypothesis remains subject to future experimental verification. Additionally, hardware implementations on FPGA platforms (e.g., Tang Primer 25K, Cyclone IV EP4CE6) and embedded microcontrollers are planned to assess *SpikeCliff* visibility in resource-constrained environments without DVFS.

As a further extension of this work, real-time visualization of *boundary-aware* timing profiles is planned as an educational and demonstrative tool, potentially implemented on a 3D LED cube platform (Raspberry Pi Pico) to illustrate timing discontinuities interactively.

References

- [1] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, Cryptographic sponge functions, <https://keccak.team/files/CSF-0.1.pdf>.
- [2] Standard for cryptographic hash functions. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, FIPS PUB 202, <https://doi.org/10.6028/NIST.FIPS.202>, [07.07.2026].
- [3] Architecture guide for Intel Turbo Boost. Intel Turbo Boost Technology 2.0 Architecture, White Paper, <https://builders.intel.com/docs/networkbuilders/intel-turbo-boost-technology-configure-per-core-turbo-overview-technology-guide-1673528561.pdf>, [07.07.2026].
- [4] Q. Ge, Y. Yarom, D. Cock, G. Heiser, A survey of microarchitectural timing attacks and countermeasures on contemporary hardware, *Journal of Cryptographic Engineering* 8(1) (2018) 1–27, <https://doi.org/10.1007/s13389-016-0141-6>.
- [5] Specification for SHA-3 derived functions. SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash, NIST Special Publication 800-185, <https://doi.org/10.6028/NIST.SP.800-185>, [07.07.2026].
- [6] J. O'Connor, J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, BLAKE3: One Function, Fast Everywhere, *IACR Cryptology ePrint Archive* 2021/1164 (2021), <https://eprint.iacr.org/2021/1164>.
- [7] J. Daemen, G. Van Assche, Differential propagation analysis of Keccak, in: *Fast Software Encryption – FSE 2012*, LNCS 7549, Springer (2012) 422–441, <https://eprint.iacr.org/2012/163>.
- [8] P. C. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, in: *Advances in Cryptology – CRYPTO 1996*, LNCS 1109, Springer (1996) 104–113, https://doi.org/10.1007/3-540-68697-5_9.
- [9] D. J. Bernstein, Cache-timing attacks on AES, Technical Report, University of Illinois at Chicago, 2005, <https://cr.yp.to/antiforgery/cachetiming-20050414.pdf>.
- [10] J. L. Hennessy, D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 6th ed., Morgan Kaufmann, 2019, ISBN 978-0-12-811905-1.
- [11] A. Kuznetsov, I. Oleshko, V. Tymchenko, K. Lisitsky, M. Rodinko, A. Kolhatin, Performance analysis of cryptographic hash functions suitable for use in blockchain, *International Journal of Computer Network and Information Security* 13(2) (2021) 1–15, <https://doi.org/10.5815/ijcnis.2021.02.01>.
- [12] S. Windarta, S. Suryadi, K. Ramli, B. Pranggono, T. S. Gunawan, Lightweight cryptographic hash functions: Design trends, comparative study, and future directions, *IEEE Access* 10 (2022) 82272–82294, <https://doi.org/10.1109/ACCESS.2022.3195572>.
- [13] I. Mufidah et al., Performance and security analysis of lightweight hash functions in IoT, *Jurnal Informatika: Jurnal Pengembangan IT* 9(3) (2024), <https://arxiv.org/pdf/2508.07840v1>.
- [14] M. Khan, D. Johansen, H. Dagenborg, A comparative analysis of lightweight hash functions using AVR ATXMEGA128 and ChipWhisperer, *arXiv preprint arXiv:2508.07840* (2025), <https://doi.org/10.48550/arXiv.2508.07840>.
- [15] Ł. Wójcik, *SpikeCliff XOF Analysis – Boundary Profiler*, GitHub repository (2026), <https://github.com/lukaszwojcikdev/spikecliff-xof-analysis/>.