Correlation Power Analysis on SHA-2-Based SLH-DSA

Jinwook Chang, Minjeong Lee, JIN YONG MIN, and Dong-Guk Han

***Abstract*—The stateless hash-based digital signature algorithm (SLH-DSA) repeatedly processes a fixed secret seed with varying public addresses. This paper presents non-profiled correlation power analysis of the address-based pseudorandom function for all six SHA-2 parameter sets, using separate recovery paths through SHA-256 compression rounds and message-schedule expansion. The procedures account for the input layout in which a 22-byte compressed address precedes the secret seed. For the 256-bit sets, a two-stage joint procedure recovers two 16-bit secret portions. We evaluate an STM32F303 ARM Cortex-M4 implementation using 30 secret seeds and 5,000 measured power traces per seed for each evaluated condition. Each recovery stage is evaluated with the required preceding secret values supplied correctly; the reported rates are therefore conditional stage-wise results, not end-to-end seed recovery rates. Compression-round stages achieve 100% success under both forest-of-random-subsets and Winternitz one-time-signature-plus address conditions. Under forest-of-random-subsets addresses, all message-schedule stages achieve 100% success for the 128- and 192-bit sets. For both 256-bit variants, joint recovery achieves 100%, whereas recovery of message word W_{12} achieves 66.7%; recovery of W_7 achieves 70.0% and 80.0% for the fast and small variants, respectively. An alternative first stage is also provided for 128-bit Winternitz one-time-signature-plus addresses. These results show that side-channel protection must cover both compression rounds and message-schedule expansion.**

***Index Terms*—Correlation power analysis, hash-based signatures, message schedule, post-quantum cryptography, SHA-256, side-channel analysis, SLH-DSA.**

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) through the Information Technology Research Center (ITRC) support program funded by the Korean government (Ministry of Science and ICT) under Grant IITP-2026-RS-2022-00164800 (50%), and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education, under Grant RS-2025-25411243 (50%).

Jinwook Chang, Minjeong Lee, and JIN YONG MIN are undergraduate students in the Department of Information Security, Cryptography and Mathematics, Kookmin University. Dong-Guk Han is a professor in the same department.

Jinwook Chang: jinwook0401@gmail.com.
ORCID: 0009-0008-5877-5761.

Minjeong Lee: min2357@kookmin.ac.kr.
ORCID: 0009-0005-4522-2189.

JIN YONG MIN: min7634@kookmin.ac.kr.
ORCID: 0009-0003-2001-8873.

Dong-Guk Han: christa@kookmin.ac.kr.
ORCID: 0000-0003-1695-5103.

I. Introduction

THE stateless hash-based digital signature algorithm (SLH-DSA), one of the three post-quantum cryptographic standards published by the National Institute of Standards and Technology (NIST) in 2024, is derived from SPHINCS+. It uses a stateless construction whose security relies on assumptions about the underlying hash functions, instantiated with either SHA-2 or SHAKE [1], [2].

Security based on hash-function assumptions does not, however, guarantee the side-channel security of a physical implementation. Power consumption, electromagnetic emissions, and execution time can disclose secret-dependent intermediate values. The pseudorandom function (PRF) of SLH-DSA repeatedly processes a fixed secret seed, `SK.seed`, together with known public addresses during signing and may therefore be susceptible to side-channel analysis.

This work analyzes the PRF of SHA-2-based SLH-DSA. We present separate secret-seed recovery procedures targeting the SHA-256 compression rounds and message schedule. Their effectiveness is evaluated using power traces acquired with a ChipWhisperer Husky from an STM32F303 ARM Cortex-M4 implementation compiled with GCC at the `-Os` optimization level.

A. Related Work

Power analysis of SHA-2 has primarily been investigated in the context of hash-based message authentication codes (HMACs). McEvoy *et al.* presented differential power analysis (DPA) and masking countermeasures for hardware implementations of HMAC based on SHA-2 [3]. Belaïd *et al.* subsequently proposed an analysis using Hamming-weight (HW) leakage [4]. Schuhmacher described the recovery of a key-derived initial state from the working-variable leakage of the outer hash [5]. Belenky *et al.* proposed a non-profiled method that exploits Hamming-distance leakage associated with addition carries [6]. Recovering these key-derived states can enable authentication-tag forgery, but it is distinct from recovering the original HMAC key.

Other studies recover secret message words directly from compression-round leakage. Park *et al.* analyzed a

fixed hardware-wallet master seed used as the message of HMAC-SHA-512 in a setting where a known, varying HMAC key changes the chaining state [7]. Wurcker and Marçais presented a sequential recovery procedure for secret message words in an HMAC setting that places a known, varying value in the key argument and a fixed secret in the message argument [8]. They evaluated correlation power analysis (CPA) of the first three secret words of HMAC-SHA-256 in simulation.

Message-schedule leakage has also been used for secret recovery. Samwel *et al.* analyzed SHA-512 in wolfSSL's deterministic Ed25519 nonce generation [9]. In the targeted input layout, a fixed auxiliary key derived from the secret key precedes the varying public message. They recovered this auxiliary key by guessing secret-dependent terms in the expanded-word relations.

Profiling-based work includes the electromagnetic template attack on HMAC-SHA-1 by Fouque *et al.* [10]. Belenky *et al.* evaluated a template attack recovering key-derived values in HMAC-SHA-2 hardware through simulation and FPGA measurements [11]. Lecomte *et al.* presented soft analytical side-channel attacks that combine message-schedule and compression-state leakage of SHA-256 and HMAC-SHA-256 in a factor graph, using simulation and electromagnetic profiling results for evaluation [12]. These profiling-based approaches differ from the non-profiled CPA setting considered here.

For hash-based signatures, Kannwischer *et al.* studied a SHA-256-based pseudorandom generator for XMSS and a BLAKE-256-based PRF for SPHINCS-256 [13]. Their SHA-2 analysis used power simulation for a particular generator construction, whereas their BLAKE-256 analysis was validated on hardware. Repeated use of the secret seed in SPHINCS+/SLH-DSA and corresponding countermeasures have also been studied. Saarinen evaluated secret-seed-dependent PRF leakage using test vector leakage assessment (TVLA) and applied masking to a Keccak/SHAKE hardware datapath [14]. Fluhrer discussed the side-channel risks of repeatedly using the same secret seed and proposed hierarchical generation of individual secret values [15].

CPA specifically targeting SHAKE-based SLH-DSA has also been proposed. Li *et al.* evaluated CPA on the θ and χ operations of the SHAKE256-based PRF using Python leakage simulations for a 128-bit parameter configuration [16]. Cao *et al.* also proposed a CPA method for SLH-DSA [17]. More recently, Carrera Rodriguez *et al.* analyzed variability in the public PRF inputs and developed a recovery procedure combining averaging of FORS (forest of random subsets) PRF subtraces that process the same target intermediate, partial-word and full-word CPA, and residual candidate search [18]. Using electromagnetic traces measured on an STM32F446RE, they demonstrated complete secret-seed recovery for all six SHAKE parameter sets.

Other studies examined power analysis of a ChaCha-based SPHINCS implementation [19], countermeasures against fault attacks on SPHINCS+ [20], and hardware resistant to timing attacks [21].

These studies show that secret inputs to SHA-2 and secret seeds in SHAKE-based SLH-DSA can be recovered from side-channel leakage. We focus on the standardized SHA-2-based SLH-DSA address PRF, where a varying 22-byte compressed address precedes the fixed `SK.seed`. We analyze the dependencies imposed by the seed length and public-address conditions, and construct recovery procedures for the SHA-256 compression rounds and message-schedule expansion. In particular, we distinguish the joint recovery of two 16-bit secrets in the 256-bit sets and the applicability of the procedures under FORS and WOTS+ (Winternitz one-time signature plus) address conditions. Each recovery stage is evaluated using measured power traces from an ARM Cortex-M4 implementation.

B. Contributions

The contributions of this work are as follows.

- We present two CPA recovery paths through the SHA-256 compression rounds and message-schedule expansion of the standardized SHA-2-based SLH-DSA address PRF on a 32-bit ARM Cortex-M4. We evaluate stage-wise recovery under FORS and WOTS+ address conditions within the applicability of each path.
- We construct message-schedule recovery procedures covering the entire `SK.seed`according to the SHA-256 block layout of each parameter set. In particular, we present a two-stage joint CPA procedure for the two 16-bit secret portions of W_5 and W_{13} in the 256-bit sets.
- We identify a difference in first-stage message-schedule recovery performance between FORS and WOTS+ address conditions and provide an alternative first stage for WOTS+ addresses in the 128-bit sets.

C. Organization

Section II describes SLH-DSA, the address PRF, the SHA-256 block layout and operations, and the CPA model. Section III defines the threat model. Section IV presents the compression-round and message-schedule recovery procedures. Section V describes the measurement environment, datasets, analysis settings, and evaluation metric. Section VI presents the experimental results. Section VII discusses countermeasures, limitations, and future work, and Section VIII concludes the paper.

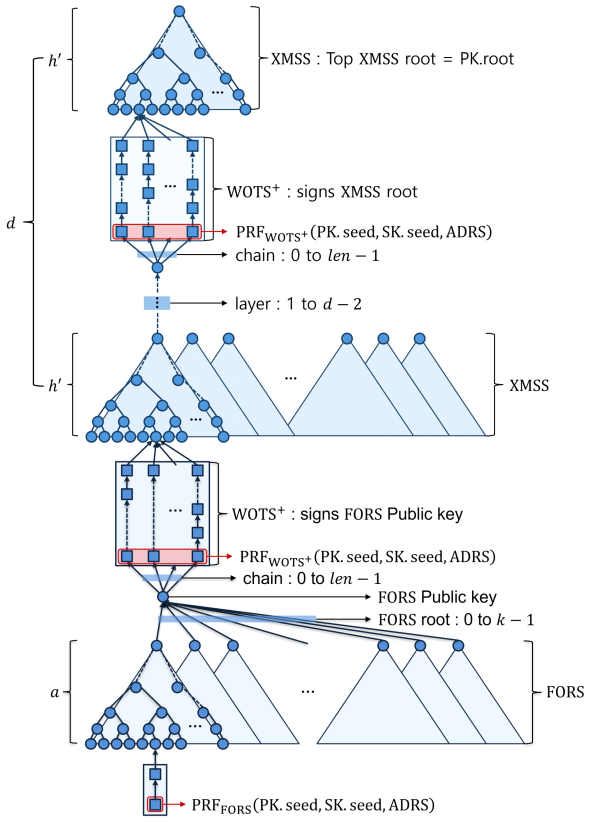


Fig. 1. SLH-DSA structure and address-PRF invocation locations. Red outlines indicate PRF invocations that generate WOTS+ chain starting values and FORS leaf secrets.

II. Background

A. SLH-DSA and the Address-Based PRF

SLH-DSA combines a hypertree with d layers of XMSS trees of height h', FORS for signing a message digest, and a collection of hash functions using the public address ADRS [1]. At each security level, the parameter sets have a fast variant, denoted by `f`, emphasizing signing speed, and a small variant, denoted by `s`, emphasizing signature size. An XMSS leaf corresponds to a WOTS+ public key, and FORS consists of k trees of height a.

WOTS+ chain starting values and FORS leaf secrets are generated as needed from the n-byte `SK.seed`using the address-based PRF. WOTS+ and FORS secret-key elements use the corresponding WOTS+ and FORS addresses. The layer, tree, type, type-specific fields, and keypair fields identify the structural position and purpose of each value. Fig. 1 shows the SLH-DSA structure and the PRF invocation locations.

When secret-key elements and authentication paths are computed according to the signing pseudocode in FIPS 205, one signature requires $k2^a$ address-PRF invocations for FORS and $d2^{h'}$len for WOTS+, where len is the number of chains in a WOTS+ instance. These invocations process the same `SK.seed`with different public addresses. These counts refer to the stated pseudocode; the actual invocation counts can differ when an implementation stores and reuses intermediate values.

B. SHA-256 Block Layout of the Address PRF

The parameter $n \in \{16, 24, 32\}$ is the secret-seed length in bytes for the selected SLH-DSA parameter set. The public seed `PK.seed`is included in both the secret and public keys.

Byte-string indices start at zero in input order, and ranges such as $\mathrm{ADRS}^{\mathrm{c}}[20{:}21]$ include both endpoints. Within a word, bit zero is the least significant bit; $W_j[b{:}a]$ includes bits a through b.

For the SHA-2 parameter sets, the address-based PRF is

$$\begin{aligned}&\mathrm{PRF}(\texttt{PK.seed}, \texttt{SK.seed}, \mathrm{ADRS})\\ &\quad = \mathrm{Trunc}_n\big(\mathrm{SHA\text{-}256}\big(\\ &\qquad \texttt{PK.seed} \,\|\, \mathrm{toByte}(0, 64-n)\\ &\qquad \|\, \mathrm{ADRS}^{\mathrm{c}} \,\|\, \texttt{SK.seed}\big)\big).\end{aligned} \tag{1}$$

where $\mathrm{ADRS}^{\mathrm{c}}$ is the 22-byte compressed address. The first 64-byte block contains only the public seed and zeros and can be precomputed. The measured final block contains $\mathrm{ADRS}^{\mathrm{c}} \,\|\, \texttt{SK.seed}$, followed by padding and the message-length encoding.

SHA-256 interprets this block as the initial words $W_0, \ldots, W_{15}$. The first five words are public. The next word is

$$W_5 = \mathrm{ADRS}^{\mathrm{c}}[20{:}21] \,\|\, \texttt{SK.seed}[0{:}1],$$

whose upper half is public and lower half is secret. The remaining secret bytes occupy consecutive words. The final two secret bytes are followed by `0x8000`. Fig. 2 illustrates the block layout.

C. SHA-256 Message Schedule and Compression Operations

For $16 \le t \le 63$, SHA-256 expands the initial block words as follows [22]:

$$\begin{aligned}W_t &= W_{t-16} + \sigma_0(W_{t-15}) + W_{t-7}\\ &\quad + \sigma_1(W_{t-2}) \pmod{2^{32}},\\ \sigma_0(x) &= \mathrm{ROTR}^7(x) \oplus \mathrm{ROTR}^{18}(x) \oplus \mathrm{SHR}^3(x),\\ \sigma_1(x) &= \mathrm{ROTR}^{17}(x) \oplus \mathrm{ROTR}^{19}(x) \oplus \mathrm{SHR}^{10}(x).\end{aligned} \tag{2}$$

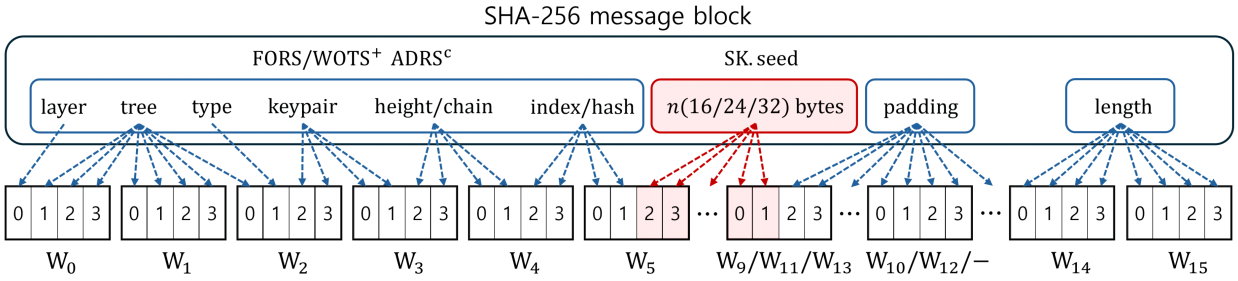


Fig. 2. Final SHA-256 block layout of the SHA-2-based SLH-DSA address PRF. Blue and red indicate public and secret regions, respectively. W_9, W_{11}, and W_{13} contain the final secret bytes for the 128-, 192-, and 256-bit seeds, respectively.

The compression rounds use

$$\begin{aligned} T_{1,t} &= H + \Sigma_1(E) + \mathrm{Ch}(E,F,G) \\ &\quad + K_t + W_t \pmod{2^{32}}, \\ T_{2,t} &= \Sigma_0(A) + \mathrm{Maj}(A,B,C) \pmod{2^{32}}, \\ \Sigma_0(x) &= \mathrm{ROTR}^2(x) \oplus \mathrm{ROTR}^{13}(x) \oplus \mathrm{ROTR}^{22}(x), \\ \Sigma_1(x) &= \mathrm{ROTR}^6(x) \oplus \mathrm{ROTR}^{11}(x) \oplus \mathrm{ROTR}^{25}(x), \\ \mathrm{Ch}(x,y,z) &= (x \wedge y) \oplus (\neg x \wedge z), \\ \mathrm{Maj}(x,y,z) &= (x \wedge y) \oplus (y \wedge z) \oplus (z \wedge x). \end{aligned} \tag{3}$$

Here, $A, \ldots, H$ denote the working variables before the round and K_t is the round constant. The updated working variable A'_t of the round using W_t is [22]

$$A'_t = T_{1,t} + T_{2,t} \pmod{2^{32}}. \tag{4}$$

D. CPA and Leakage Model

For a candidate k, known public input X_i, and target intermediate $f(X_i, k)$, CPA uses the HW hypothesis

$$H_i(k) = \mathrm{HW}\big(f(X_i, k)\big). \tag{5}$$

The candidate score is the maximum absolute Pearson correlation between $H_i(k)$ and the measured traces over the time samples [23].

Ordinary word-recovery stages use cumulative-HW CPA with 8-bit partitions [24]. For a 32-bit guessed variable, candidates are extended from the least significant byte upward. The HW of the lower output bytes determined by a candidate path is added to the HW of the current output byte to form the leakage hypothesis. The carry into the upper byte is also computed from that candidate path. Known public input bytes are fixed, and only unknown bytes are guessed.

At each partition, a candidate is scored by the maximum absolute correlation between its hypothesis and the measured traces. A path score is the sum of the partition scores accumulated so far, and the best B_{byte} paths are retained for the next partition. After the last partition, the path with the highest score is selected. Cumulative HW constructs an individual leakage hypothesis, whereas summing path scores combines the assessments of multiple partitions; these are separate operations.

The alternative first stage for 128-bit WOTS+ addresses exhaustively evaluates a 16-bit secret. Joint recovery of W_5/W_{13} in the 256-bit sets combines exhaustive first-stage class evaluation with second-stage byte-wise search. These procedures are described in Sections IV-C and IV-B, respectively.

III. Threat Model

We consider a non-profiled adversary with physical access to a device performing SLH-DSA signing or key generation with a fixed secret key. The adversary can measure the device's power consumption and knows the parameter set, implementation, and public key, but does not know `SK.seed`and cannot use profiling. The signing addresses are determined by the message, the randomizer R included in the signature, and the resulting public indices. The adversary can therefore recompute the addresses of the PRF invocations from the message and signature.

Rather than executing a complete signature, our experiment invokes the PRF individually with valid public addresses and records the address associated with each power trace. This evaluates PRF computations with known public inputs; it does not evaluate invocation identification or segmentation within complete signing traces. The datasets and address-generation procedure are described in Section V-B.

IV. Secret-Seed Recovery Procedures

A. Compression-Round Path

The compression-round attack uses the public SHA-256 state derived from `PK.seed`and the public words $W_0, \ldots, W_4$ to compute the working state immediately before the first secret-bearing round. Candidates for the secret portion of W_5 are substituted into (4), and the predicted HW of A'_t is used as the CPA hypothesis. Each word is recovered using the 8-bit partitioned procedure

in Section II-D. The recovered word updates the working state for the next secret-bearing round. This process continues through W_9, W_{11}, or W_{13} for the 128-, 192-, or 256-bit sets, respectively. If all stages succeed, concatenating the recovered secret portions in their original order reconstructs the complete `SK.seed`.

B. Message-Schedule Path

This subsection describes the recovery procedure under FORS address conditions. The alternative first stage for WOTS+ addresses is presented in Section IV-C. At each security level, recovery begins with expanded words appearing early in the message schedule. Secret-bearing initial words are recovered through the expanded-word relations, and the recovered values supply the words needed for subsequent stages.

For W_6, W_7, and W_8, the guessed variables are $Y_j = \sigma_0(W_j)$, $j \in \{6, 7, 8\}$, rather than the original words. The corresponding relations are

$$\begin{aligned} W_{21} &= W_5 + Y_6 + W_{14} + \sigma_1(W_{19}), \\ W_{22} &= W_6 + Y_7 + W_{15} + \sigma_1(W_{20}), \\ W_{23} &= W_7 + Y_8 + W_{16} + \sigma_1(W_{21}), \end{aligned} \tag{6}$$

where all additions are modulo 2^{32}. In each relation, the terms other than Y_j are computed from public values and preceding recovery results. The expanded word is thus expressed as the sum of known terms and Y_j, which is recovered in 8-bit partitions from the expanded-word leakage. SHA-256's σ_0 is an invertible 32-bit linear transformation over GF(2). Once all 32 bits of Y_j have been recovered, a precomputed inverse reconstructs the original word as $W_j = \sigma_0^{-1}(Y_j)$. The result table identifies the original recovered word W_j, although the actual search variable in these stages is Y_j.

1) 128-bit parameter sets: The secret upper 16 bits of W_9 are recovered using

$$W_{16} = W_0 + \sigma_0(W_1) + W_9 + \sigma_1(W_{14}) \pmod{2^{32}}. \tag{7}$$

The lower 16 bits are public padding, so the secret domain contains 2^{16} values. Next, the secret portion of W_5 is recovered from

$$W_{20} = W_4 + \sigma_0(W_5) + W_{13} + \sigma_1(W_{18}) \pmod{2^{32}}. \tag{8}$$

Write $W_5 = P_5 \parallel S_5$, where P_5 and S_5 are its public upper and secret lower 16-bit halves. Linearity of σ_0 gives

$$\sigma_0(W_5) = \sigma_0(P_5 \parallel 0^{16}) \oplus \sigma_0(0^{16} \parallel S_5). \tag{9}$$

The term $\sigma_0(P_5 \parallel 0^{16})$ is computable for each trace from the known public address. The fixed transformed value $Y_5 = \sigma_0(0^{16} \parallel S_5)$ is used as the guessed variable.

Although Y_5 has a 32-bit representation, only 2^{16} transformed values are attainable because the original secret S_5 is 16 bits long. The implementation does not impose this attainable-set constraint during candidate generation or selection. Instead, it guesses the four bytes of Y_5 from least to most significant using 8-bit partitioned CPA. Only the best B_{byte} candidate paths survive each partition, so this is not an exhaustive search over all 2^{32} values. Substituting (9) into (8) constructs the leakage hypothesis for W_{20}. A recovery is counted as successful only when all 32 bits of the selected Y_5 match the reference transformed value. Given the correct Y_5, the lower 16 bits of $\sigma_0^{-1}(Y_5)$ yield S_5. The same W_5 recovery method is used for the 192-bit sets.

After W_5, the words W_6, W_7, and W_8 are recovered from W_{21}, W_{22}, and W_{23}, respectively. The recovery order for the 128-bit sets is therefore

$$W_9 \rightarrow W_5 \rightarrow W_6 \rightarrow W_7 \rightarrow W_8.$$

2) 192-bit parameter sets: The expanded words W_{16}, W_{17}, and W_{18} recover W_9, W_{10}, and W_{11}, respectively. All 32 bits of W_9 and W_{10} are secret. Only the upper 16 bits of W_{11} are secret; its lower half is the public padding `0x8000`. The secret portion of W_5 is then recovered by separating the public contribution and guessing the transformed secret contribution as in the 128-bit case. Finally, W_{21}, W_{22}, and W_{23} recover W_6, W_7, and W_8. The resulting order is

$$W_9 \rightarrow W_{10} \rightarrow W_{11} \rightarrow W_5 \rightarrow W_6 \rightarrow W_7 \rightarrow W_8.$$

3) 256-bit parameter sets: The words W_9 through W_{12} are recovered from W_{16} through W_{19}. With $W_5 = P_5 \parallel S_5$ and $W_{13} = S_{13} \parallel$ `0x8000`, relation (8) contains the two secret values S_5 and S_{13}. Unlike the W_5 recovery for the 128- and 192-bit sets, it therefore requires the following two-stage joint analysis.

A direct exhaustive search would require 2^{32} candidates. To reduce the search, Stage 1 evaluates the 2^{15} classes defined by the bits of S_5 other than $S_5[2]$ using

$$H_{1,i}(c) = \mathrm{HW}\left(W_{20,i}^{(c)}[15{:}0]\right). \tag{10}$$

The best B_{class} classes are retained. Here, c indexes the candidate class and B_{class} is the number of classes passed to Stage 2. The grouping is possible because $S_5[2]$ does not affect the lower 16 output bits used in Stage 1. Fig. 3 illustrates the lower- and upper-half dependencies.

In Stage 2, each retained class is expanded for both values of $S_5[2]$, yielding complete 16-bit S_5 candidates. For each such candidate, the lower and upper bytes of S_{13} are guessed in sequence. These occupy bits [23:16] and [31:24] of W_{13}, respectively. Each byte partition retains the best B_{byte} paths, and the byte search is performed separately for each S_5 candidate.

Let s denote a 16-bit candidate for S_5, and let u and v be candidates for the lower and upper bytes of S_{13}. Thus, the S_{13} candidate is $u + 2^8 v$, and the corresponding

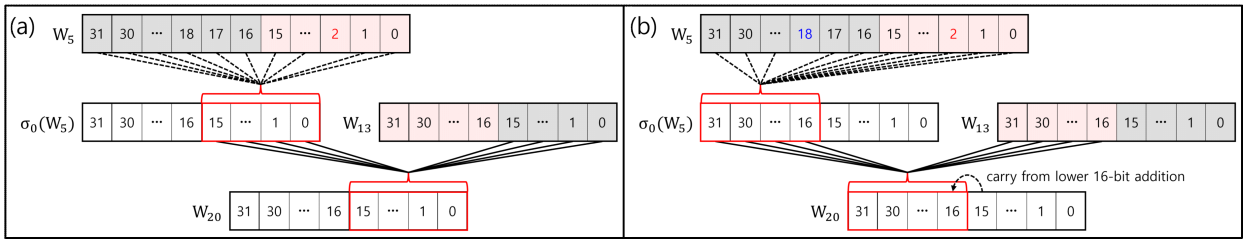


Fig. 3. Bit dependencies for joint W_5/W_{13} recovery in the 256-bit sets. Light pink and gray denote secret and public inputs, respectively; red outlines highlight the bit ranges being illustrated. (a) Dependencies of the lower 16 bits of W_{20}. (b) Dependencies of the upper 16 bits of W_{20}, including carry propagation.

W_{13} candidate is $v \parallel u \parallel$ `0x8000`. Let $W_{20,i}^{(s,u,v)}$ be the intermediate obtained by substituting the public input of trace i and the candidates (s,u,v) into (8). The two byte-partition hypotheses are

$$H_{2,\text{low},i}(s,u) = \text{HW}\left(W_{20,i}^{(s,u,0)}[23{:}0]\right), \quad H_{2,\text{high},i}(s,u,v) = \text{HW}_{32}\left(W_{20,i}^{(s,u,v)}\right). \tag{11}$$

The first partition uses the cumulative HW of the lower 24 bits, so the still-unknown upper byte v does not affect that hypothesis. The second combines a retained u candidate with a new v candidate and computes the HW of the full 32-bit output. Both hypotheses include the lower output bytes and the effects of carries in modular addition.

The final score of a candidate pair is the sum of its Stage 1 class score and its two byte-partition scores. The pair with the highest combined score among all surviving pairs is selected. Recovery then proceeds to W_6, W_7, and W_8 through W_{21}, W_{22}, and W_{23}.

C. Alternative First Stage for WOTS+ Addresses

The W_{16}-based first stage used for FORS addresses exhibited reduced recovery performance under WOTS+ address conditions. For the 128-bit sets, we therefore use the leakage of W_{18} as an alternative:

$$W_{18} = W_2 + \sigma_0(W_3) + W_{11} + \sigma_1(W_{16}) \pmod{2^{32}}. \tag{12}$$

In the 128-bit block, W_{11} is public padding, and the lower 16 bits of W_9 are also fixed to `0x8000`. The recovery target is consequently the secret upper half $S_9 = W_9[31{:}16]$.

All 2^{16} values are guessed and evaluated without byte partitioning. For each 16-bit candidate g, we construct $W_9^{(g)} = g \parallel$ `0x8000`, compute $W_{16}^{(g)}$ from the public input using (7), and substitute it into (12) to obtain $W_{18}^{(g)}$.

The leakage hypothesis is the full 32-bit HW of $W_{18}^{(g)}$. Its maximum absolute correlation with the measured traces is the candidate score, and the highest-scoring value among all 2^{16} candidates is selected to recover W_9. This procedure uses neither 8-bit candidate expansion nor a sum of partition scores. After W_9, the recovery of W_5 through W_8 is identical to that under FORS address conditions.

For the 192- and 256-bit blocks, W_{11} contains secret-seed bytes. Applying the same alternative therefore requires an additional secret-value assumption or a separate recovery procedure. This paper does not provide a complete message-schedule recovery procedure for WOTS+ addresses in these two seed-length families.

V. Experimental Setup

A. Target Device and Measurement Environment

Experiments were performed on an STM32F303 ARM Cortex-M4 mounted on a CW308 UFO board, using the SHA-2-based SLH-DSA reference implementation [25]. Power traces were acquired with a ChipWhisperer Husky, without additional preprocessing. The measurement window was configured to cover the SHA-256 compression rounds and message-schedule expansion. The trigger was asserted immediately before the first compression round and deasserted after the last round. The window therefore included 64 compression rounds and three message-schedule expansion operations.

For each build configuration, the trace length was set automatically from the trigger duration measured during calibration. No pretrigger samples or additional guard samples were used. CPA was applied over the complete recorded interval of each trace. Fig. 4 and Table I summarize the equipment and settings.

B. Datasets and Address Sampling

For each parameter set and address type, we use 30 experimental `SK.seed` values and analyze 5,000 measured power traces per seed together with the corresponding public addresses. Within a parameter-set/address condition, the same public-address sequence is reused across seed experiments to reduce differences in the public-input samples when comparing seeds.

For WOTS+ addresses, the layer is sampled from its valid range, after which the tree address is generated from the range associated with that layer. The keypair and chain fields are sampled from their respective valid

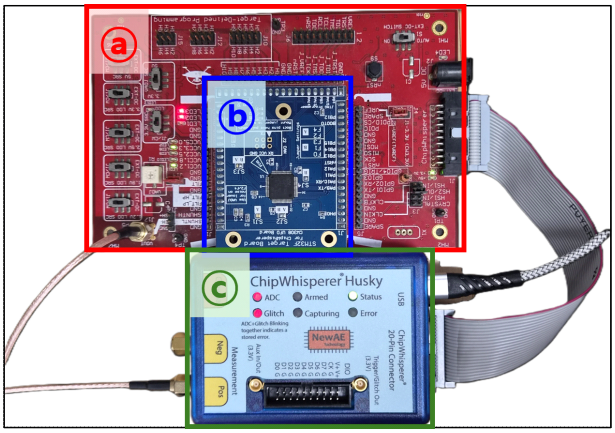


Fig. 4. Experimental setup for power-trace acquisition: (a) CW308 UFO board, (b) target board equipped with an STM32F303 microcontroller, and (c) ChipWhisperer Husky.

TABLE I
EXPERIMENTAL EQUIPMENT AND MEASUREMENT/IMPLEMENTATION SETTINGS

Parameter	**Configuration**
Target platform	CW308 UFO
Target MCU	STM32F303
Acquisition device	ChipWhisperer Husky
Target clock frequency	7.37 MHz
ADC sampling rate	29.48 MS/s (4×)
ADC resolution	12-bit
Measurement gain	15 dB
Compiler / optimization	GCC / `-Os`
Trigger region	SHA-256 compression rounds and message-schedule expansion

ranges. For FORS addresses, the layer and tree height are fixed to zero, and the tree, keypair, and tree-index fields are sampled from their valid ranges. This procedure evaluates PRF leakage for valid public inputs rather than reproducing the invocation order of an actual signature. It is not uniform sampling over the entire space of valid addresses, and duplicate addresses can occur.

The secret seeds are generated for each byte length n by a deterministic pseudorandom generator with a fixed master seed. The fast and small variants sharing a seed length use the same set of 30 seeds, and these seed sets are reused across builds. These values support reproducibility and ground-truth validation; they are not a proposed key-generation method for operational use.

C. CPA Settings and Evaluation Procedure

We select the cumulative-HW model based on compression-round recovery performance and apply it consistently to both the compression-round and message-schedule paths. Byte-partition hypotheses follow the cumulative-HW model in Section II-D. Carries are incorporated through 32-bit modular addition, but the HW of the carry bits is not included as a separate leakage term.

Each word-recovery or joint-recovery stage is evaluated independently, conditional on the preceding secret values required by its hypotheses being correct. Within an individual word recovered by 8-bit partitioned CPA, however, lower-byte candidates are propagated to the next partition. Candidate paths are therefore chained within a word, whereas recovery errors do not propagate between independently evaluated words. The ground truth of the current target is used to assess success and candidate ranks, not to select a candidate.

The number of retained paths at each byte partition is $B_{\text{byte}} = 4$. For joint recovery in the 256-bit sets, the number of retained Stage 1 classes is $B_{\text{class}} = 4$. These are separate settings for byte-wise candidate paths and joint-recovery classes. The W_{18}-based first stage for 128-bit WOTS+ exhaustively evaluates the 2^{16} candidates for the secret upper half of W_9 and therefore uses neither setting.

Final candidates in ordinary partitioned CPA are ranked by the path score defined in Section II-D. The W_{18}-based recovery and the W_5/W_{13} joint recovery use the candidate scores defined in Sections IV-C and IV-B, respectively.

D. Recovery Success Rate

Recovery succeeds when the final selected candidate matches the ground truth of the stage. When an initial message word is guessed directly, only its secret portion is compared; public address bytes and padding are excluded. When the guessed variable is a transformed value Y_j, the entire transformed value must match. Joint W_5/W_{13} recovery succeeds only when both secret portions are correct. A case is counted as a failure if the correct candidate is pruned or is not ultimately selected. The resulting success rates describe conditional stage-wise performance with correct preceding secret values, rather than end-to-end recovery of `SK.seed`.

Let N_s be the number of seed experiments evaluated at stage s, let $k^*_{s,j}$ be the correct secret value for seed j, and let $\hat{k}_{s,j}$ be the final highest-scoring candidate selected using 5,000 traces. The stage-wise success rate (SR) is

$$\mathrm{SR}_s = \frac{1}{N_s} \sum_{j=1}^{N_s} \mathbf{1}\left[\hat{k}_{s,j} = k^*_{s,j}\right], \tag{13}$$

where $\mathbf{1}[\cdot]$ is one if the condition holds and zero otherwise.

For joint recovery, the Stage 1 retention rate of the correct class is recorded separately from the final recovery success rate. Retention succeeds if the class containing the correct S_5 is among the best B_{class} classes. Retaining that class alone does not constitute a successful joint recovery:

both secret values in the final selected pair must match. Cases in which the correct class is eliminated in Stage 1 are included among final recovery failures.

VI. Experimental Results

We apply the procedure in Section V-C to the datasets in Section V-B and report stage-wise recovery performance using the success criterion in Section V-D.

A. Compression-Round CPA

The upper part of Table II reports stage-wise recovery using the updated state A'_t as the leakage intermediate. Each stage recovers the secret portion of W_t. The evaluated words range from W_5 to W_9 for the 128-bit sets, from W_5 to W_{11} for the 192-bit sets, and from W_5 to W_{13} for the 256-bit sets.

Every evaluated stage achieves 100% success under both FORS and WOTS+ address conditions for all six parameter sets. In other words, each stage correctly recovers its target secret portion for all 30 evaluated seeds. No success-rate difference is observed between the fast and small variants with the same seed length. These results show that the A'_t hypothesis distinguishes secret-word candidates under both address conditions in this experimental setting.

B. Message-Schedule CPA

The lower part of Table II reports stage-wise recovery using expanded message-schedule words as leakage intermediates. For the 128-bit sets, all stages recovering W_9 and W_5 through W_8 achieve 100% success under both FORS and WOTS+ addresses. For the 192-bit sets under FORS addresses, recovery of W_9 through W_{11} and the subsequent recovery of W_5 through W_8 also achieve 100% success.

The 256-bit sets under FORS addresses show stage-dependent performance. Recovery of W_9, W_{10}, and W_{11} from W_{16}, W_{17}, and W_{18}, respectively, achieves 100% success for both variants. In contrast, recovery of W_{12} from W_{19} succeeds for 20 of 30 seeds, or 66.7%, in each variant.

Joint recovery of W_5/W_{13} from W_{20} achieves 100% success (30/30) for both variants. Both secret portions are therefore recovered correctly for every evaluated seed in each condition. Recovery of W_6 from W_{21} likewise achieves 100% success for both variants.

Recovery of W_7 from W_{22} succeeds in 70.0% of cases (21/30) for 256f and 80.0% (24/30) for 256s. Recovery of W_8 from W_{23} reaches 100% for both. Thus, the reduced success rates in the cumulative-HW message-schedule evaluation are concentrated in the W_{12} and W_7 stages of the 256-bit sets.

Although W_{12} and W_7 do not achieve 100% success, their recovery errors are not propagated to the independently evaluated dependent stages. The required preceding secrets are supplied correctly at each stage. Consequently, a W_{12} error is not injected into the evaluation of W_6, which uses W_{19}, and a W_7 error is not injected into the evaluation of W_8. The 100% rates of those later stages must not be interpreted as evidence that they correct preceding recovery errors.

No complete message-schedule recovery procedure is provided here for WOTS+ addresses in the 192- and 256-bit families. Those conditions are consequently not evaluated in the stage-wise table and are marked with a dash. The dash does not mean that their success rate is zero.

Overall, the compression-round path exhibits consistent stage-wise recovery across the evaluated conditions, whereas some message-schedule stages in the 256-bit sets show reduced success rates. These rates alone do not establish the causes of the difference. Further examination of correct-candidate ranks and score gaps to major incorrect candidates is needed for the affected stages.

VII. Discussion

A. Countermeasures

Properly implemented masking can mitigate first-order leakage from secret-dependent intermediates and has been applied to the Keccak/SHAKE datapath of an SLH-DSA hardware accelerator [14]. Applying masking in SHA-256 software requires consideration of implementation complexity, randomness requirements, and the performance cost of protecting arithmetic in both compression rounds and the message schedule. Hierarchical generation of individual secret values is another possible approach to reducing direct repeated use of the same seed [15]. Although such a construction is compatible with the standard verification procedure, it does not preserve the standard key-generation and signing procedures unchanged, so standard conformance must also be considered.

A hiding approach is to randomize the order of PRF invocations where their required addresses and inputs are already available. Outputs can be associated with their addresses and consumed by subsequent computations, while preserving dependencies in the WOTS+ chains and tree operations. The required buffering must be considered. Randomizing invocation order can make it harder to associate PRF calls with addresses in a complete signing trace, but it does not remove leakage within each PRF invocation. Its effectiveness must therefore be evaluated together with the possibility of reidentifying and aligning calls and the associated memory and execution-time costs. These countermeasures are not experimentally evaluated in this work; their validation is left to future research.

TABLE II
STAGE-WISE RECOVERY SUCCESS RATES FOR COMPRESSION-ROUND AND MESSAGE-SCHEDULE CPA

Compression-Round CPA							
`SK.seed` length (bits)	Recovery step	Recovered secret	Leakage intermediate	**Recovery success rate (%)**			
				FORS		WOTS+	
				f	s	f	s
128	1–5	$W_5, \ldots, W_9$	$A'_5, \ldots, A'_9$	100	100	100	100
192	1–7	$W_5, \ldots, W_{11}$	$A'_5, \ldots, A'_{11}$	100	100	100	100
256	1–9	$W_5, \ldots, W_{13}$	$A'_5, \ldots, A'_{13}$	100	100	100	100
Message-Schedule CPA							
`SK.seed` length (bits)	Recovery step	Recovered secret	Leakage intermediate	**Recovery success rate (%)**			
				FORS		WOTS+	
				f	s	f	s
128	1	W_9	W_{16} / W_{18}[a]	100	100	100	100
	2–5	$W_5, \ldots, W_8$	$W_{20}, \ldots, W_{23}$	100	100	100	100
192	1–3	$W_9, \ldots, W_{11}$	$W_{16}, \ldots, W_{18}$	100	100	—	—
	4–7	$W_5, \ldots, W_8$	$W_{20}, \ldots, W_{23}$	100	100	—	—
256	1	W_9	W_{16}	100	100	—	—
	2	W_{10}	W_{17}	100	100	—	—
	3	W_{11}	W_{18}	100	100	—	—
	4	W_{12}	W_{19}	66.7	66.7	—	—
	5	(W_5, W_{13})	W_{20}	100	100	—	—
	6	W_6	W_{21}	100	100	—	—
	7	W_7	W_{22}	70.0	80.0	—	—
	8	W_8	W_{23}	100	100	—	—

[a] For the 128-bit first stage, W_{16} is used for FORS and W_{18} for WOTS+.
Each stage is evaluated with its required preceding secret values supplied correctly, using 30 seeds and 5,000 traces per seed for each condition. A grouped row indicates that every included stage has the displayed success rate. W_t denotes the secret portion of that word. Joint (W_5, W_{13}) recovery requires both secret portions to be correct. A dash indicates a condition not evaluated here; it does not denote a zero success rate.

B. Limitations and Future Work

The proposed procedures construct later leakage hypotheses from preceding recovered values. Selecting an incorrect candidate can therefore affect stages that depend on that value. The stage-wise evaluation in Section V-C excludes such error propagation. Assessing the reliability of the complete procedure requires an additional chained evaluation that propagates actual recovery outputs. The affected stages depend on the data dependencies; a failure does not necessarily cause every subsequent stage to fail.

This paper does not provide a complete message-schedule recovery procedure for WOTS+ addresses in the 192- and 256-bit sets. This is a limitation of the evaluated relations and search procedures, not evidence that those conditions are inherently resistant to power analysis. Other intermediates and candidate-validation methods that reduce the need to guess additional secrets simultaneously should be investigated.

The experiment measures individual PRF invocations with valid addresses rather than complete signature executions. It therefore does not fully capture the invocation order and address dependencies in actual signing. Future work should identify and segment PRF calls in complete signing traces before applying the proposed procedures. In addition, candidate scores are computed over the complete measurement window of an individual PRF trace. Mapping the observed correlation peaks to actual instructions and intermediate values remains necessary for a more detailed leakage interpretation.

Future work will compare recovery performance and computational cost as the number of retained candidates and the candidate-reevaluation strategy change. We also plan to evaluate the security benefits and implementation costs of hierarchical secret-value generation and hiding through randomized PRF invocation order.

VIII. CONCLUSION

This paper presented separate CPA procedures targeting the SHA-256 compression rounds and message-schedule expansion of the address-based PRF in SHA-2-based SLH-DSA on an STM32F303 ARM Cortex-M4. The procedures account for the parameter-dependent block layout in which a varying public compressed address precedes the fixed `SK.seed`. We evaluated each recovery stage using measured power traces with the required preceding secret values supplied correctly.

All evaluated stages of the compression-round path achieved 100% success for every parameter set and

address condition, whereas reduced success rates were observed for some message-schedule stages in the 256-bit sets. In particular, we constructed a two-stage joint CPA procedure that avoids directly evaluating all 2^{32} secret pairs. These findings show that side-channel protection of SHA-2-based SLH-DSA must cover message-schedule expansion as well as compression rounds. The reliability of complete seed recovery nevertheless requires further evaluation that propagates actual recovery outputs from one stage to the next.

References

[1] National Institute of Standards and Technology, *Stateless Hash-Based Digital Signature Standard*, Aug. 2024, FIPS PUB 205, doi: 10.6028/NIST.FIPS.205.

[2] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The SPHINCS+ signature framework," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2019, pp. 2129–2146, doi: 10.1145/3319535.3363229.

[3] R. McEvoy, M. Tunstall, C. C. Murphy, and W. P. Marnane, "Differential power analysis of HMAC based on SHA-2, and countermeasures," in *Information Security Applications (WISA 2007)*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer, 2007, vol. 4867, pp. 317–332, doi: 10.1007/978-3-540-77535-5_23.

[4] S. Belaïd, L. Bettale, E. Dottax, L. Genelle, and F. Rondepierre, "Differential power analysis of HMAC SHA-2 in the Hamming weight model," in *Proc. Int. Conf. Secur. Cryptogr. (SECRYPT)*, 2013, pp. 230–241, doi: 10.5220/0004532702300241.

[5] F. Schuhmacher, "Canonical DPA attack on HMAC-SHA1/SHA2," in *Constructive Side-Channel Analysis and Secure Design (COSADE 2022)*, ser. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2022, vol. 13211, pp. 193–211, doi: 10.1007/978-3-030-99766-3_9.

[6] Y. Belenky *et al.*, "Carry-based differential power analysis (CDPA) and its application to attacking HMAC-SHA-2," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2023, no. 3, pp. 1–29, 2023, doi: 10.46586/tches.v2023.i3.1-29.

[7] D. Park, J. Kim, H. Kim, and S. Hong, "Cloning hardware wallet without valid credentials through side-channel analysis of hash function," *IEEE Access*, vol. 12, pp. 132 677–132 688, 2024, doi: 10.1109/ACCESS.2024.3440370.

[8] A. Wurcker and D. Marçais, "The dangerous message/key swap in HMAC," in *Constructive Approaches for Security Analysis and Design of Embedded Systems (CASCADE 2025)*, ser. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2026, vol. 15952, pp. 291–310, doi: 10.1007/978-3-032-01405-4_12.

[9] N. Samwel, L. Batina, G. Bertoni, J. Daemen, and R. Susella, "Breaking Ed25519 in WolfSSL," in *Topics in Cryptology (CT-RSA 2018)*, ser. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2018, vol. 10808, pp. 1–20, doi: 10.1007/978-3-319-76953-0_1.

[10] P.-A. Fouque, G. Leurent, D. Réal, and F. Valette, "Practical electromagnetic template attack on HMAC," in *Cryptographic Hardware and Embedded Systems (CHES 2009)*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer, 2009, vol. 5747, pp. 66–80, doi: 10.1007/978-3-642-04138-9_6.

[11] Y. Belenky, I. Dushar, V. Teper, H. Chernyshchyk, L. Azriel, and Y. Kreimer, "First full-fledged side channel attack on HMAC-SHA-2," in *Constructive Side-Channel Analysis and Secure Design (COSADE 2021)*, ser. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2021, vol. 12910, pp. 31–52, doi: 10.1007/978-3-030-89915-8_2.

[12] M. Lecomte, J. Maillard, A. Moran, G. Renault, and B. Smith, "Soft analytical side-channel attacks on SHA-2 and HMAC," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2026, no. 3, pp. 1205–1227, 2026, doi: 10.46586/tches.v2026.i3.1205-1227.

[13] M. J. Kannwischer, A. Genêt, D. Butin, J. Krämer, and J. Buchmann, "Differential power analysis of XMSS and SPHINCS," in *Constructive Side-Channel Analysis and Secure Design (COSADE 2018)*, ser. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2018, vol. 10815, pp. 168–188, doi: 10.1007/978-3-319-89641-0_10.

[14] M.-J. O. Saarinen, "Accelerating SLH-DSA by two orders of magnitude with a single hash unit," in *Advances in Cryptology (CRYPTO 2024)*, ser. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2024, vol. 14920, pp. 276–304, doi: 10.1007/978-3-031-68376-3_9.

[15] S. Fluhrer, "Side channel resistant SPHINCS+," Cryptology ePrint Archive, Paper 2024/500, 2024.

[16] X. Li, Y. Cao, Z. Du, F. Zhang, and J. Peng, "Correlation power analysis against the secret key seed of SLH-DSA," in *Modern Management Based on Big Data VII*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, 2026, vol. 424, pp. 648–653, doi: 10.3233/FAIA260630.

[17] Y. Cao, J. Li, Z. Du, K. Yang, and H. Li, "Correlation power analysis on SLH-DSA algorithm," in *Modern Management Based on Big Data VII*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, 2026, vol. 424, pp. 553–558.

[18] R. Carrera Rodriguez, J. Zahreddine, A. Monthieu, A. Vasselle, and P.-Y. Liardet, "SHAKEing SLH-DSA with non-profiled SCAs," Cryptology ePrint Archive, Paper 2026/2584, 2026.

[19] V. Satheesh and D. Shanmugam, "Implementation vulnerability analysis: A case study on ChaCha of SPHINCS," in *Proc. IEEE Int. Symp. Smart Electron. Syst. (iSES)*, 2020, pp. 97–102, doi: 10.1109/iSES50453.2020.00032.

[20] A. Genêt, "On protecting SPHINCS+ against fault attacks," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2023, no. 2, pp. 80–114, 2023, doi: 10.46586/tches.v2023.i2.80-114.

[21] T. Huang *et al.*, "A timing attack resistant lightweight post-quantum crypto-processor for SPHINCS+," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2024, pp. 1–5, doi: 10.1109/ISCAS58744.2024.10558654.

[22] National Institute of Standards and Technology, *Secure Hash Standard (SHS)*, Aug. 2015, FIPS PUB 180-4, doi: 10.6028/NIST.FIPS.180-4.

[23] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems (CHES 2004)*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer, 2004, vol. 3156, pp. 16–29, doi: 10.1007/978-3-540-28632-5_2.

[24] M. Tunstall, N. Hanley, R. P. McEvoy, C. Whelan, C. C. Murphy, and W. P. Marnane, "Correlation power analysis of large word sizes," in *Proc. IET Irish Signals Syst. Conf. (ISSC)*, 2007, pp. 145–150.

[25] SPHINCS+ Project, "SPHINCS+ reference implementation, FIPS 205 branch (`bas/fips205`)," GitHub repository, commit `ef36d20`, accessed: May 15, 2026. [Online]. Available: https://github.com/sphincs/sphincsplus/tree/bas/fips205