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

Jinwook Chang, Minjeong Lee, Jin Yong Min, and Dong-Guk Han

***Abstract*—SLH-DSA repeatedly processes a fixed secret seed together with varying public addresses, making its physical implementation a target for side-channel analysis. We present non-profiled correlation power analysis (CPA) procedures for the address-based pseudorandom function in all six SHA-2-based SLH-DSA parameter sets, using SHA-256 compression rounds and message-schedule expansion. The procedures include joint recovery of two partial secrets in the 256-bit parameter sets and an alternative first message-schedule step for 128-bit WOTS+ addresses. Using power traces measured on an STM32F303, we evaluate conditional, stage-wise recovery with 30 seeds and 5,000 traces per seed. All compression-round stages achieve a 100% success rate, whereas some message-schedule stages in the 256-bit sets achieve 66.7–80.0%. In chained evaluations using addresses derived from a single signature, WOTS+ compression-round CPA recovers the entire secret seed in all nine cases for each parameter set. Message-schedule CPA also succeeds for 128-bit WOTS+ addresses, whereas full recovery fails under FORS addresses. Compression-round CPA on key-generation paths achieves full recovery in 11 of 18 cases. These evaluations use individually measured PRF executions and are distinct from analyses of continuous signature or key-generation traces. Finally, we define four scopes for shuffling initial PRF calls and theoretically compare their permutation counts, random-index selection counts, and PRF-output buffer requirements.**

***Index Terms*—SLH-DSA, SHA-256, correlation power analysis, side-channel analysis, hash-based signatures, shuffling.**

I. Introduction

The Stateless Hash-Based Digital Signature Algorithm (SLH-DSA) is one of three post-quantum standards released by the National Institute of Standards and Technology (NIST) in 2024. Derived from SPHINCS+, it relies on hash-function security assumptions and uses SHA-2 or SHAKE [1], [2].

Hash-based post-quantum security does not ensure physical side-channel security: power, electromagnetic, and timing leakage can reveal secret-dependent intermediates. The SLH-DSA pseudorandom function (PRF) repeatedly processes the fixed `SK.seed` with known public addresses during signing, making it a potential target.

We study this address PRF through SHA-256 compression-round and message-schedule leakage, using ChipWhisperer Husky power measurements of an STM32F303 ARM Cortex-M4 implementation compiled with GCC `-Os`.

A. Related Work

SHA-2 power analysis has primarily considered hash-based message authentication codes (HMACs). McEvoy et al. presented differential power analysis (DPA) and masking for HMAC-SHA-2 hardware [3]; Belaïd et al. studied Hamming-weight (HW) leakage [4]. Schuhmacher recovered key-derived initial states from outer-hash working-variable leakage [5], and Belenky et al. exploited carry-dependent Hamming-distance (HD) leakage in non-profiled analysis [6]. Such state recovery can enable tag forgery without recovering the original HMAC key.

Compression-round leakage can also expose secret message words. Park et al. analyzed a fixed hardware-wallet master seed supplied as an HMAC-SHA-512 message, with known, varying keys changing the chaining state [7]. Wurcker and Marçais similarly considered known, varying HMAC keys and fixed secret messages [8]. They described sequential message-word recovery and simulated correlation power analysis (CPA) of the first three secret words of HMAC-SHA-256.

Samwel et al. used message-schedule leakage in SHA-512 for wolfSSL's deterministic Ed25519 nonce generation [9]. A fixed, private-key-derived auxiliary key precedes the varying public message; secret-dependent terms in expanded-word relations enabled its recovery.

Profiling-based work includes Fouque et al.'s electromagnetic template attack on HMAC-SHA-1 [10] and Belenky et al.'s template attack on key-derived values in HMAC-SHA-2 hardware, evaluated through simulation and FPGA measurements [11]. Lecomte et al. combined message-schedule and compression-state leakage in factor graphs for soft analytical side-channel attacks (SASCA) on SHA-256 and HMAC-SHA-256, using simulation and electromagnetic profiling [12]. These profiling assumptions differ from ours.

The authors are with the Department of Information Security, Cryptography and Mathematics, Kookmin University (e-mail: jinwook0401@gmail.com; min2357@kookmin.ac.kr; min7634@kookmin.ac.kr; christa@kookmin.ac.kr).

Kannwischer et al. analyzed a SHA-256-based XMSS pseudorandom generator using power simulation for that construction, and validated an analysis of the SPHINCS-256 BLAKE-256 PRF on hardware [13].

For SPHINCS+/SLH-DSA, Saarinen assessed secret-seed-dependent PRF leakage using test vector leakage assessment (TVLA) and masked a Keccak/SHAKE hardware path [14]. Fluhrer addressed repeated secret-seed use through hierarchical generation of individual secrets [15].

For SHAKE-based SLH-DSA, Li et al. targeted the SHAKE256 address PRF's θ and χ operations to recover `SK.seed` in Python leakage simulation of a 128-bit setting [16]. Cao et al. targeted `SK.prf` in $\mathrm{PRF}_{\mathrm{msg}}$, which generates the signing randomizer R [17]. Using column-parity leakage from SHAKE256's first θ operation, they recovered the 128-bit secret from 50 simulated traces with Gaussian noise, without device measurements. Carrera Rodriguez et al. analyzed public-input variability and combined averaging of FORS PRF subtraces sharing the target intermediate, partial-word and full-word CPA, and remaining-candidate enumeration [18]. STM32F446RE electromagnetic measurements demonstrated full seed recovery for all six SHAKE parameter sets. Other work covers ChaCha-based SPHINCS power analysis [19], SPHINCS+ fault-attack countermeasures [20], and timing-attack-resistant hardware [21].

These studies concern either the SHAKE256 address PRF and `SK.seed` [16], [18] or $\mathrm{PRF}_{\mathrm{msg}}$ and `SK.prf` [17]. Their results do not establish recovery performance for SHA-2 implementations, which require SHA-256-specific procedures and measured evaluation.

We study the standard layout placing a fixed `SK.seed` after a varying 22-byte compressed address, distinguishing FORS and WOTS+ (Winternitz One-Time Signature Plus) conditions. Stage-wise evaluations assume the required preceding secrets are correct.

B. Contributions

Using measured STM32F303 ARM Cortex-M4 traces for all six SHA-2 parameter sets, we make four contributions:

- A secret-seed recovery procedure using SHA-256 compression-round intermediates, with conditional stage-wise evaluation under FORS and WOTS+ addresses across all six parameter sets.
- Parameter-specific message-schedule recovery procedures, including a two-stage joint CPA for the 16-bit partial secrets in W_5 and W_{13} of the 256-bit sets. Observed first-stage differences between FORS and WOTS+ motivate an alternative path for 128-bit WOTS+ addresses.
- Chained full-seed recovery using addresses from a single signature path, succeeding for WOTS+ compression-round CPA in all six sets and WOTS+ message-schedule CPA in the 128-bit sets. Additional compression-round evaluations use key-generation WOTS+ addresses. Both use separately executed PRF calls, not continuous signing or key-generation traces.
- Four shuffling scopes—Local, Subtree, Hierarchical, and Global—for the FIPS 205 PRF-call set, with theoretical comparisons of permutation counts, random-index selections, and PRF-output buffer requirements.

II. Background

A. SLH-DSA and Its Address-Based PRF

SLH-DSA combines d layers of height-h' XMSS trees, a forest of random subsets (FORS) signing the message digest, and hash functions using public addresses `ADRS` [1]. At each security level, the `f` (fast) and `s` (small) variants prioritize signing speed and signature size, respectively. XMSS leaves correspond to WOTS+ public keys; FORS contains k height-a trees.

The address PRF generates WOTS+ chain starting values and FORS leaf secrets as needed from the n-byte `SK.seed`, using their respective address types. Layer, tree, type-dependent, and key-pair fields identify each value's position and purpose. Figure 1 shows the construction and PRF-call locations.

Computing secret-key elements and authentication paths according to the FIPS 205 signing pseudocode requires $k2^a$ FORS and $d2^{h'}\mathrm{len}$ WOTS+ address-PRF calls per signature, where len is the WOTS+ chain count. They share `SK.seed` but use different public addresses. Implementations storing and reusing intermediates may have different call counts.

Key generation constructs the topmost XMSS tree for `PK.root`, requiring $2^{h'}\mathrm{len}$ PRF calls for its WOTS+ secret-key elements under the standard pseudocode [1]. Their address set is message-independent.

B. SHA-256 Block Layout of the Address PRF

The security parameter $n \in \{16, 24, 32\}$ specifies the seed length in bytes. The public value `PK.seed` is a component of both the private and public keys. Byte indices start at zero in input order; intervals such as $\mathtt{ADRS}^c[20{:}21]$ include both endpoints. Word bits start at zero at the least significant bit, and $W_j[b{:}a]$ includes bits a through b.

For SHA-2 parameter sets, the address PRF is

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

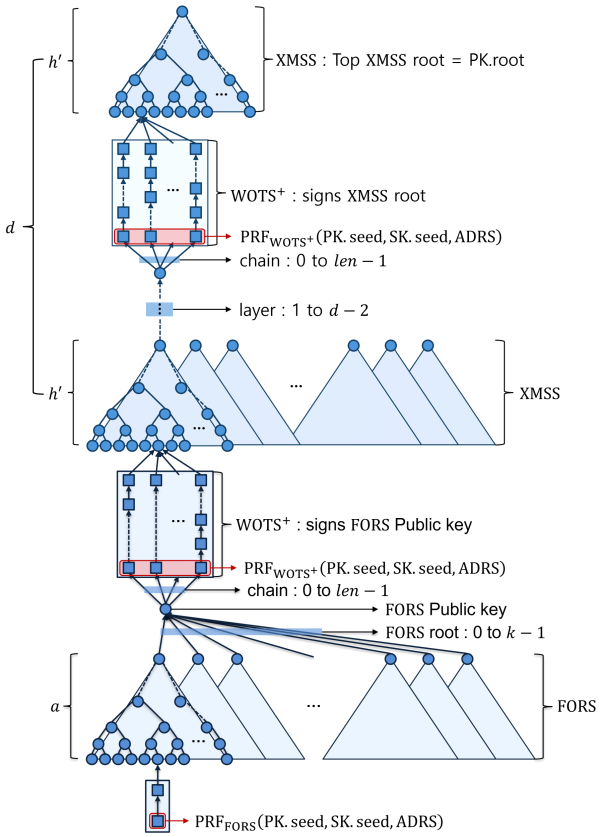


Fig. 1. SLH-DSA structure and PRF-call locations. Red outlines identify the calls that generate WOTS+ chain starting values and FORS leaf secrets.

where $\texttt{ADRS}^c$ is the 22-byte compressed address. The first 64-byte block, containing the public seed and zeros, can be precomputed. The measured block contains $\texttt{ADRS}^c \parallel \texttt{SK.seed}$, padding, and the encoded message length.

The block provides initial words $W_0, \ldots, W_{15}$. Words $W_0, \ldots, W_4$ are public; $W_5 = \texttt{ADRS}^c[20{:}21] \parallel \texttt{SK.seed}[0{:}1]$ has a public upper half and a secret lower half. Subsequent words contain the remaining secret bytes, with `0x8000` following the last two. Figure 2 shows the layout.

C. SHA-256 Message Schedule and Compression

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

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

The compression function uses

$$\begin{aligned} T_{1,t} &= H_t + \Sigma_1(E_t) + \mathrm{Ch}(E_t, F_t, G_t) \\ &\quad + K_t + W_t \pmod{2^{32}}, \\ T_{2,t} &= \Sigma_0(A_t) + \mathrm{Maj}(A_t, B_t, C_t) \pmod{2^{32}}. \\ \Sigma_0(x) &= \mathrm{ROTR}^2(x) \oplus \mathrm{ROTR}^{13}(x) \\ &\quad \oplus \mathrm{ROTR}^{22}(x), \\ \Sigma_1(x) &= \mathrm{ROTR}^6(x) \oplus \mathrm{ROTR}^{11}(x) \\ &\quad \oplus \mathrm{ROTR}^{25}(x), \\ \mathrm{Ch}(x,y,z) &= (x \wedge y) \oplus (\neg x \wedge z), \\ \mathrm{Maj}(x,y,z) &= (x \wedge y) \oplus (x \wedge z) \\ &\quad \oplus (y \wedge z). \end{aligned} \tag{3}$$

Round t uses W_t and updates working variable A to [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 $f(X_i, k)$, the HW leakage hypothesis is

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

The candidate score is the maximum absolute Pearson correlation between this hypothesis and the measured traces over all time samples [23].

Ordinary word recovery uses 8-bit partitioned CPA with cumulative HW [24]. A 32-bit candidate is extended from its least significant byte. Each hypothesis sums the HW of the current output byte and lower output bytes implied by the candidate path. Carries are computed along that path and reflected in modular-addition results, without adding carry-bit HW separately. Public bytes remain fixed; only unknown bytes are hypothesized.

Each partition uses maximum absolute correlation as its score. Scores are summed along each path, retaining the top B_{byte} paths for the next partition and selecting the highest-scoring path at the end. Thus, cumulative HW forms one leakage hypothesis, whereas score summation combines successive partitions.

The first 128-bit WOTS+ stage evaluates a full 16-bit secret at once; 256-bit joint W_5/W_{13} recovery combines exhaustive candidate-class evaluation with byte-wise search. Sections IV-C and IV-B describe these procedures, respectively.

III. Threat Model

We assume a non-profiled adversary who can observe device power consumption during SLH-DSA signing or key generation. Each recovery case uses PRF traces with the same `SK.seed` and `PK.seed`. The adversary knows the parameter set, implementation, and public key, but not `SK.seed`, and does not use profiling.

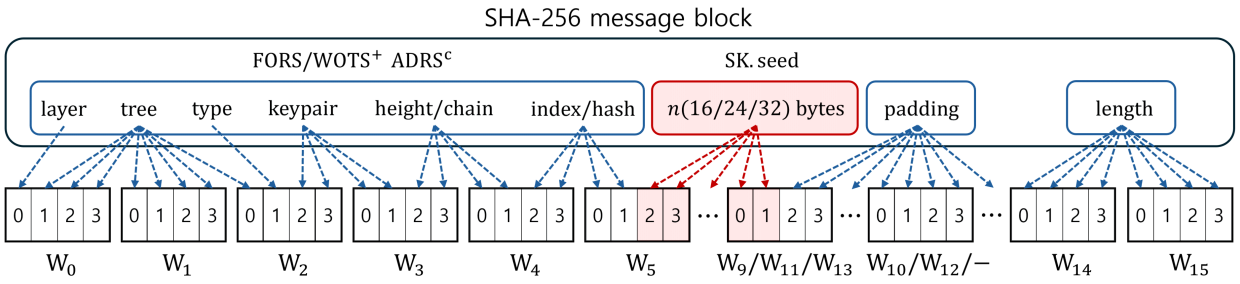


Fig. 2. Final SHA-256 block of the SHA-2-based SLH-DSA address PRF. Blue and red denote public and secret regions, respectively. Words W_9, W_{11}, and W_{13} contain the last secret bytes for 128-, 192-, and 256-bit seeds, respectively.

The addresses used during signing are determined by the message, the randomizer R included in the signature, and the resulting public indices. The adversary can therefore recompute all PRF-call addresses from the message and signature. Key-generation PRF addresses are determined by the public structure of the topmost XMSS tree and do not depend on a message. Exploiting key-generation leakage requires access to power measurements while that operation is executed.

Our experiments invoke the PRF separately with valid public addresses and record the address corresponding to each power trace, rather than collecting a continuous execution of the entire signing or key-generation function. They evaluate PRF computations with known public inputs and do not validate identification and segmentation of individual calls in continuous signature or key-generation traces. Section V-B describes the datasets and address-generation methods.

IV. Secret-Seed Recovery Procedures

A. Compression-Round Path

The compression-round path uses A'_t in (4) as the leakage intermediate. The public seed `PK.seed` and public message words determine the state preceding the first secret-containing word. Candidate-dependent intermediates are compared with measured traces.

Each word uses the cumulative-HW, 8-bit partitioned CPA of Section II-D. Public bits are fixed according to Section II-B, and secret-bit candidates are evaluated. Each recovered word updates the compression state for the next stage.

Recovery proceeds from W_5 to W_9, W_{11}, or W_{13} for the 128-, 192-, or 256-bit sets, respectively. Concatenating the recovered seed bytes in their original order reconstructs `SK.seed`. This chains actual recovered values; Section V-C distinguishes conditional evaluation with correct preceding secrets.

B. Message-Schedule Path

This subsection describes the message-schedule procedure under FORS addresses; the alternative first stage for WOTS+ addresses is given in Section IV-C. Recovery starts from early expanded words and uses the schedule relations to recover words containing secret bits. The recovered values determine the words needed by 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:

$$\begin{aligned} W_{21} &= W_5 + Y_6 + W_{14} + \sigma_1(W_{19}), \\ W_{22} &= W_6 + Y_7 + W_{15} + \sigma_1(W_{20}), \pmod{2^{32}}. \\ W_{23} &= W_7 + Y_8 + W_{16} + \sigma_1(W_{21}) \end{aligned} \tag{6}$$

Public inputs and preceding recovered values determine every term except Y_j, which is recovered in 8-bit partitions from the expanded word's leakage. Since σ_0 is an invertible 32-bit linear transformation over $\mathrm{GF}(2)$, its precomputed inverse gives $W_j = \sigma_0^{-1}(Y_j)$ after all 32 bits are recovered. Tables identify the original W_j, although the searched variable is Y_j.

a) 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, giving 2^{16} candidates. Next, the partial secret in W_5 is recovered through

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

Write $W_5 = P_5 \parallel S_5$, where P_5 and S_5 are the public upper and secret lower 16 bits. Linearity gives

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

The address-dependent term $\sigma_0(P_5 \| 0^{16})$ can be computed for each trace. The guessed variable is therefore $Y_5 = \sigma_0(0^{16} \parallel S_5)$, which is constant for a fixed secret seed.

Although Y_5 is represented by 32 bits, only 2^{16} transformed values are valid because S_5 has 16 bits. The implementation does not impose this validity constraint when generating or filtering candidates. It performs 8-bit partitioned CPA over the four bytes of Y_5, starting with the least significant byte and retaining the best B_{byte} candidate paths at each partition. This is not an

exhaustive search over all 2^{32} candidates. Substitution into (8) defines the leakage hypotheses for W_{20}. Success requires the selected 32-bit value of Y_5 to match the correct transformed value. Applying σ_0^{-1} then yields S_5 in the lower 16 bits. The same W_5 procedure applies to the 192-bit parameter sets.

After W_5, words W_6, W_7, and W_8 are recovered using W_{21}, W_{22}, and W_{23}, respectively. The recovery order is consequently $W_9 \to W_5 \to W_6 \to W_7 \to W_8$.

b) 192-bit parameter sets.: Words W_{16}, W_{17}, and W_{18} yield W_9, W_{10}, and W_{11}. The first two are entirely secret; W_{11} has 16 secret upper bits and public lower padding `0x8000`. The preceding procedure recovers W_5 via its transformed secret component, followed by W_6, W_7, and W_8 through W_{21}, W_{22}, and W_{23}. The order is $W_9 \to W_{10} \to W_{11} \to W_5 \to W_6 \to W_7 \to W_8$.

c) 256-bit parameter sets.: Words W_9 through W_{12} are recovered from W_{16} through W_{19}. Together with $W_5 = P_5 \parallel S_5$, let $W_{13} = S_{13} \parallel \texttt{0x8000}$. Equation (8) now contains both secrets S_5 and S_{13}, requiring the following two-stage joint analysis.

A direct search would require 2^{32} candidates. The first stage instead evaluates 2^{15} candidate classes, defined by the 15 bits of S_5 other than $S_5[2]$, and retains the best B_{class} classes. With class index c, the hypothesis is

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

Bit $S_5[2]$ does not affect the lower 16 output bits used at this stage. Figure 3 shows the input dependencies of the lower 16 bits and of the upper 16 bits including carry propagation.

The second stage expands each retained class with the two values of $S_5[2]$ to obtain complete 16-bit candidates for S_5. For each such candidate, the lower and upper bytes of S_{13} are guessed in sequence, corresponding to bits $[23:16]$ and $[31:24]$ of W_{13}. Each partition retains the best B_{byte} paths, and byte searches for different S_5 candidates are performed independently.

Let s be a candidate for S_5 and let u and v be candidates for the lower and upper bytes of S_{13}. Thus $S_{13} = u + 2^8 v$ and the candidate word is $W_{13} = v \parallel u \parallel \texttt{0x8000}$. Denote by $W_{20,i}^{(s,u,v)}$ the intermediate obtained from (8) using trace i's public input and these candidates. The hypotheses are

$$\begin{aligned} H_{2,\mathrm{low},i}(s,u) &= \mathrm{HW}\Big(W_{20,i}^{(s,u,0)}[23:0]\Big), \\ H_{2,\mathrm{high},i}(s,u,v) &= \mathrm{HW}_{32}\Big(W_{20,i}^{(s,u,v)}\Big). \end{aligned} \tag{11}$$

The first partition uses cumulative HW over the lower 24 output bits, so the as-yet unguessed upper byte v does not affect this hypothesis. The second partition combines retained u candidates with new v candidates and uses the HW of all 32 bits. Both hypotheses include the lower output bytes and the effects of carries in modular addition.

The first-stage class score and two byte-partition scores are summed for each pair. The best retained pair is selected, followed by recovery of 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 under FORS addresses exhibits reduced recovery performance under WOTS+ addresses. For the 128-bit parameter sets, we therefore consider the leakage of W_{18}:

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

In the 128-bit block, W_{11} is public padding and the lower 16 bits of W_9 are fixed to `0x8000`. The target is $S_9 = W_9[31:16]$, for which all 2^{16} candidates are evaluated together.

For each g, set $W_9^{(g)} = g \parallel \texttt{0x8000}$ and use the public input with (7) and (12) to compute $W_{16}^{(g)}$ and $W_{18}^{(g)}$. The hypothesis is the 32-bit HW of $W_{18}^{(g)}$, scored by its maximum absolute correlation with measured traces. The best of all 2^{16} candidates is selected without bytewise expansion or partition-score summation. Subsequent W_5–W_8 recovery follows the FORS procedure.

For the 192- and 256-bit blocks, W_{11} contains secret seed bits. Applying the same alternative would require assumptions about additional secret values or a separate recovery procedure. We do not present a complete message-schedule-only procedure for WOTS+ addresses in those two parameter families.

V. Experimental Setup

A. Target and Measurement Setup

The SHA-2-based SLH-DSA reference implementation [25] runs on an STM32F303 ARM Cortex-M4 mounted on a CW308 UFO board. A ChipWhisperer Husky acquires traces without additional preprocessing. The trigger spans the first through last SHA-256 compression rounds, capturing all 64 rounds and three message-schedule expansion phases. Trace length follows each build's calibrated trigger duration, without pretrigger samples or additional margin; CPA uses the entire recording. Figure 4 and Table I summarize the setup.

B. Datasets and Address Sampling

a) Random-address dataset.: For each parameter set and address type, the baseline uses 30 experimental values of `SK.seed`, with 5,000 power traces and their corresponding public addresses per seed. Within a parameter/address condition, the same address sequence is reused across seeds to reduce differences in public-input samples.

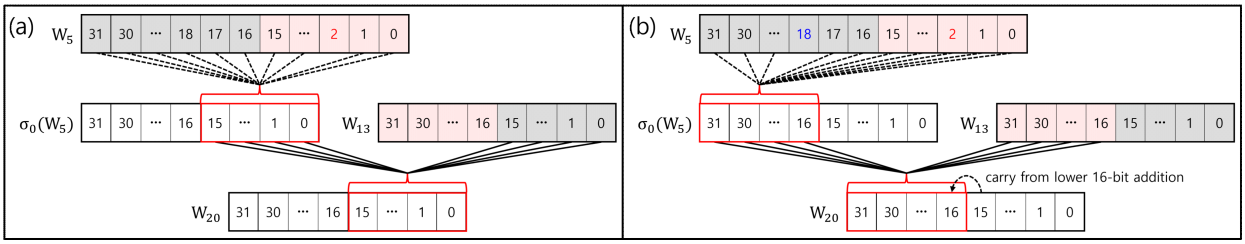


Fig. 3. Bit dependencies for joint recovery of W_5/W_{13} in the 256-bit parameter sets. Light pink and gray denote secret and public inputs, respectively; red outlines mark the output bits of interest. (a) Dependencies of the lower 16 bits of W_{20}. (b) Dependencies of the upper 16 bits, including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

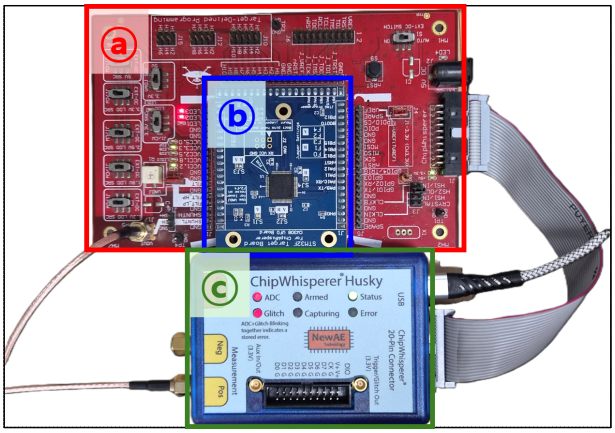


Fig. 4. Power-measurement setup: (a) CW308 UFO board, (b) target board carrying the STM32F303, and (c) ChipWhisperer Husky.

TABLE I
MEASUREMENT EQUIPMENT AND IMPLEMENTATION SETTINGS.

Parameter	Configuration
Target platform	CW308 UFO
Target MCU	STM32F303
Acquisition device	ChipWhisperer Husky
Target clock	7.37 MHz
ADC sampling rate	29.48 MS/s (4×)
ADC resolution	12 bits
Measurement gain	15 dB
Compiler / optimization	GCC / `-Os`

WOTS+ sampling selects a valid layer, generates its tree address, and samples valid key-pair and chain fields. FORS sampling fixes the layer and tree height to zero and samples valid tree, key-pair, and tree-index fields. This evaluates leakage for valid public inputs rather than reproducing a signing sequence. Sampling is not necessarily uniform over the valid address space, and addresses may repeat.

A deterministic pseudorandom generator with a fixed master seed generates the experimental seeds of each length n. Matching f and s sets share the same 30 seeds. This supports reproducibility and ground-truth verification, not operational key generation.

b) Signature-path dataset.: For each parameter set, three distinct seeds and three signing paths per seed give nine seed–signature cases. The host computes each path's public addresses, and the target records the corresponding PRFs individually. The recordings preserve address composition and call order, but are not continuous full-signing traces.

Each analysis uses all traces of one address type from one signing path, without mixing FORS/WOTS+ or different signatures. One signature entails multiple PRF calls and therefore multiple traces. Reported counts follow the acquisition-file records.

c) Key-generation-path dataset.: Another dataset uses the initial WOTS+ PRF addresses invoked when computing the public-key root of the topmost XMSS tree. Three distinct values of `SK.seed` for each of the six SHA-2 parameter sets give 18 parameter–seed cases. For each seed, the address set of one key-generation path is used, and each address PRF is executed individually on the target. Both `SK.seed` and `PK.seed` remain fixed within a dataset. These traces are not combined with signing-path traces. Section VI-E reports the trace counts and complete-seed recovery results.

C. CPA Configuration and Evaluation Procedure

The cumulative-HW model, candidate scores, and path scores of Section II-D are used for byte-partitioned CPA in both paths. In the random-address baseline, each word-recovery or joint-recovery stage is evaluated independently with the correct preceding secret values required by its hypotheses. Within a word processed by 8-bit partitioned CPA, however, lower-byte candidates are passed to the next partition. Candidate paths are therefore linked within a word, while errors between different words are not propagated. The target's ground truth is used to evaluate success and rank, not to select candidates.

The number of paths retained at each byte partition is $B_{\text{byte}} = 4$, and the first stage of the 256-bit joint recovery retains $B_{\text{class}} = 4$ classes. These are separate parameters

governing byte paths and joint-recovery classes. Neither applies to the 128-bit WOTS+ W_{18}-based stage, which evaluates all 2^{16} candidates for the upper secret half of W_9. Final ranks in ordinary partitioned CPA follow the path scores of Section II-D; the W_{18}-based and joint procedures use the scores specified in Sections IV-C and IV-B, respectively.

Signature-path evaluation analyzes FORS and WOTS+ separately and passes actual selected values to subsequent stages without supplying correct preceding secrets. Success requires a full `SK.seed` match. Ground truth verifies results but does not select candidates or replace recovered values. Each of a seed's three signatures is analyzed independently, with no transfer of recovered values between signatures. The compression-round and message-schedule paths are likewise separate, without combining their outputs.

Key-generation evaluation applies compression-round CPA to each seed's WOTS+ PRF traces, chaining actual recovered values without supplying preceding ground truth for selection or hypotheses. Success requires a complete seed match. Message-schedule CPA is not evaluated for these paths.

D. Evaluation Metrics

1) Stagewise Recovery Success Rate: A stage succeeds when its final selected candidate matches the ground truth. For directly guessed message words, only secret bits are checked, excluding public address and padding bits. For transformed variables Y_j, the entire transformed value must match. Joint recovery of W_5/W_{13} succeeds only when both partial secrets match. Elimination of the correct candidate during the search or failure to select it at the end counts as failure. This baseline metric is conditional stagewise performance, distinct from the chained recovery of the complete seed in the path-based experiments.

Let N_s be the number of independent seeds evaluated at stage s, $k^*_{s,j}$ the correct secret for seed j, and $\widehat{k}_{s,j}$ the highest-scoring candidate using 5,000 traces. The success rate is

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

where $\mathbf{1}[\cdot]$ is one if its argument is true and zero otherwise.

Correct first-stage class retention is recorded separately: the class containing S_5 must be among the best B_{class} classes. Final success still requires both selected secrets to match; eliminating the correct class counts as failure.

2) Complete Recovery on Signing and Key-Generation Paths: For the one-signature condition, complete recovery is evaluated separately by parameter set, address type, and recovery path. Let j index a seed, r index a signature for that seed, S_j be the true seed, and $\widehat{S}_{j,r}$ be the seed reconstructed using only that signature. Define

$$I_{j,r} = \mathbf{1}\left[\widehat{S}_{j,r} = S_j\right], \qquad j, r \in \{1,2,3\}. \tag{14}$$

When all nine cases of a condition have been analyzed, its complete recovery rate is

$$\mathrm{SR}^{(1)}_{\text{full}} = \frac{1}{9}\sum_{j=1}^{3}\sum_{r=1}^{3} I_{j,r}. \tag{15}$$

We report successful-case counts and rates. Per-seed outcomes count successful signatures out of three. Since signatures share their seed, the nine cases do not represent nine distinct secrets.

For key generation, the rate is the fraction of three distinct seeds per parameter set completely recovered from one path each.

3) Candidate Rank and Guessing Entropy: Rank evaluation covers three targets in each of 128f and 128s: FORS compression-round recovery of W_5, FORS recovery of W_9 using W_{16}, and WOTS+ recovery of W_9 using W_{18}. For 256f and 256s, it evaluates the first-stage classes and second-stage pairs of FORS joint W_5/W_{13} recovery. Each condition averages results over 30 seeds. Guessing entropy (GE) is used for the eight conditions ranked over the entire candidate space defined for that target. The two second-stage conditions use a penalized mean rank because they involve class restriction and elimination.

For GE, let $r_{s,j}(m)$ be the correct candidate's rank for target s and seed j, computed from the first m traces. Then

$$\mathrm{GE}_s(m) = \frac{1}{N_s}\sum_{j=1}^{N_s} r_{s,j}(m), \qquad m \in M. \tag{16}$$

Ranks start at one: the rank equals one plus the number of candidates whose scores exceed the correct candidate's score by more than 10^{-6}. Consequently, GE equal to one follows this ranking rule and does not imply a uniquely highest-scoring correct candidate or successful actual candidate selection.

The rank evaluation uses cumulative HW but differs from the recovery search. For 128-bit FORS compression-round W_5 recovery and W_{16}-based W_9 recovery, the 16-bit secret is split into two bytes. Their partition scores are summed to rank all 2^{16} candidates, without the stagewise candidate retention and elimination used in actual recovery. For WOTS+ W_{18}-based W_9 recovery, each 16-bit candidate is substituted into W_{16} and W_{18}, and the HW of all 32 bits of W_{18} determines its score. All 2^{16} scores are compared to obtain the rank.

For the first stage of 256-bit joint recovery, the 2^{15} classes defined by the bits of S_5 other than bit 2 are ranked. At each trace count, the second-stage evaluation

selects the best four first-stage classes anew and expands each with both values of $S_5[2]$. Combining each resulting S_5 candidate with the 2^{16} candidates for S_{13} gives 524,288 pairs. This evaluation retains first-stage class selection but does not apply the second stage's internal bytewise pruning.

If the correct class survives stage one, the correct pair is ranked among the 524,288 selected pairs. Otherwise, it is assigned 524,289. The mean of these values across 30 seeds is the *penalized mean rank*. It combines ranking within a restricted set with a penalty for class elimination, and is distinct from GE over all 2^{32} pairs.

Both rank metrics are evaluated at every trace count from 3 to 500 and then at intervals of 250. One trace does not define Pearson correlation; with two traces, nonconstant sample vectors have absolute correlation one, which is unsuitable for this ranking. The evaluation set is

$$M = \{3, 4, \ldots, 500\} \cup \{750, 1000, 1250, \ldots, 5000\}. \tag{17}$$

At each m, the first m traces in acquisition order are used. Smaller GE places the correct candidate nearer the top of its ranking. Metrics for different candidate spaces and ranking rules are reported separately rather than pooled across conditions.

Write $Q_s(m)$ for the mean-rank metric of target s. It equals $\mathrm{GE}_s(m)$ in eight conditions and the penalized mean rank in the two joint second-stage conditions. We distinguish the first occurrence of one from the point after which one is maintained at all evaluated counts:

$$m_s^{\mathrm{stab}} = \min\{m \in M : Q_s(m') = 1 \ \forall m' \in M,\ m' \geq m\}. \tag{18}$$

Stable N denotes m_s^{stab} and concerns only evaluated counts through 5,000 traces; spacing by 250 above 500 establishes nothing at intermediate counts. A target's mean rank of one is not complete chained `SK.seed` recovery or a guarantee for all keys.

VI. Experimental Results

Using Sections V-B–V-C, we report conditional stage-wise recovery and rank metrics for the random-address baseline, followed by complete-seed recovery on individual signing and key-generation paths.

A. Compression-Round CPA

Table II reports A'_t-based recovery of the secret bits in each target word.

Every stage achieves 100% success across all 30 seeds, both address types, and all six parameter sets, with no f/s difference. Under these conditions, A'_t-based hypotheses distinguish the target candidates.

B. Message-Schedule CPA

Message-schedule CPA achieves 100% in every evaluated stage for the 128-bit sets under both address types and the 192-bit sets under FORS addresses (Table II).

The 256-bit FORS results vary across stages. Recovery of W_9, W_{10}, and W_{11} using W_{16}, W_{17}, and W_{18} achieves 100% in both variants. Recovery of W_{12} using W_{19} achieves 66.7% (20/30) in both. Joint recovery of W_5/W_{13} through W_{20} achieves 100% (30/30), meaning that both partial secrets match for every evaluated seed. Recovery of W_6 through W_{21} also achieves 100%.

Recovery of W_7 through W_{22} achieves 70.0% (21/30) for 256f and 80.0% (24/30) for 256s. Recovery of W_8 through W_{23} achieves 100% in both. The reduced success rates in this cumulative-HW evaluation are thus concentrated in W_{12} and W_7 of the 256-bit sets.

These failures are not propagated to separately evaluated dependent stages. Correct preceding values are supplied for each hypothesis, so the W_6 evaluation, which depends on W_{19}, does not inherit a W_{12} error; likewise, a W_7 error does not enter the W_8 evaluation. A later 100% rate does not mean that an earlier error has been corrected.

Dashes denote unevaluated 192- and 256-bit WOTS+ message-schedule conditions, for which no complete procedure is presented, rather than 0% success. The observed contrast with compression-round results does not establish the causes of performance differences or individual failures.

C. Candidate-Rank Evaluation

Table III reports the ten conditions defined in Section V-D3: eight GE and two joint second-stage penalized-mean-rank conditions. Figure 5 plots four. Each metric averages 30 random-address seeds with correct preceding secrets supplied.

For 128-bit FORS compression-round W_5 recovery, GE exceeds 20,000 at 100 traces but remains one from 1,000 traces onward for both variants. Figure 5(a) shows 128f. The sustained-rank-one counts for W_{16}-based W_9 recovery are 394 and 1,250 for 128f and 128s; panel (b) shows 128f. For WOTS+ W_{18}-based W_9 recovery, they are 96 and 86, respectively, with 128f shown in panel (c).

For the 256-bit FORS joint recovery, first-stage GE remains one from 385 and 376 traces for 256f and 256s. Second-stage penalized mean rank remains one from 245 and 270 traces; panel (d) shows 256f. The second stage can evaluate the correct class whenever it is retained, even if it is not ranked first in the first stage. The ordering of these onset counts is therefore consistent. The metrics have different candidate sets and elimination rules, so their magnitudes or onset counts alone do not compare the difficulty of the two stages.

TABLE II

STAGEWISE RECOVERY SUCCESS RATES FOR COMPRESSION-ROUND AND MESSAGE-SCHEDULE CPA. EACH CONDITION USES 30 SEEDS AND 5,000 TRACES PER SEED, WITH THE CORRECT PRECEDING SECRET VALUES SUPPLIED. W_t IDENTIFIES A MESSAGE WORD CONTAINING THE TARGET SECRET BITS. ALL STAGES GROUPED IN A ROW HAVE THE REPORTED RATE. A DASH INDICATES NO REPORTED RESULT FOR THAT ROW/ADDRESS COMBINATION; THE 192- AND 256-BIT WOTS+ MESSAGE-SCHEDULE CONDITIONS WERE NOT EVALUATED.

Seed length (bits)	Step	Target word(s)	Leakage intermediate	FORS (%)		WOTS+ (%)	
				f	s	f	s
Compression-round CPA							
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							
128	1	W_9	W_{16} (FORS)	100	100	—	—
			W_{18} (WOTS+)	—	—	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	—	—

At 5,000 traces, all eight GE conditions and both penalized-mean-rank conditions equal one. This follows the ranking rules in Section V-D3 and does not replace the success rate of the actual pruned search or of complete chained seed recovery. The representative targets do not include the 256-bit W_{12} and W_7 stages with reduced success in Section VI-B; these curves consequently do not explain those failures.

D. Recovery Along a Single Signing Path

Table IV summarizes nine independently analyzed cases per parameter set, using the single-signature chained evaluation of Section V-C.

In Table IV, compression-round CPA recovers the entire seed in all nine WOTS+ cases for every parameter set, but in no FORS case. Message-schedule CPA also succeeds in every 128-bit WOTS+ case for both variants, while complete recovery fails under FORS addresses for all parameter sets. The 192- and 256-bit WOTS+ message-schedule-only conditions are not evaluated.

All six FORS conditions fail at the first target: W_5 for compression rounds and W_9 for the message schedule. These errors affect subsequent hypotheses; complete-seed failure therefore does not establish independent failure at every later word.

The random-address conditional rates and signature-path complete-recovery rates concern different conditions. Besides address generation, trace counts and the provision of preceding secrets differ. Their rate differences cannot therefore be attributed solely to address distributions. The observed FORS/WOTS+ contrast in this one-signature setting calls for separate analysis of the first-stage failures and input conditions.

These experiments analyze individually executed PRFs whose addresses follow a single signing path, not a continuous trace of the complete signing function. The results concern the evaluated seeds and signatures, without guaranteeing success for all keys and messages. Estimating the minimum number of signatures by pooling different signatures is outside the scope of this study.

E. Recovery Along a Key-Generation Path

Table V reports compression-round CPA on one key-generation path per seed, using the chained evaluation of Section V-C.

Complete-seed recovery succeeds in 11 of 18 parameter–seed cases (Table V).

All three seeds fail at the first target W_5 for 128f and 192f. For 256f, one seed has a W_{11} recovery error followed by mismatches at later stages. Since an earlier error affects subsequent hypotheses, these later mismatches are not all independent failures.

These results demonstrate seed recovery for address sets used during key generation as well as signing. Parameter sets differ in both PRF call counts and public-address composition, so differences in recovery rates cannot be attributed to trace count alone.

TABLE III

MEAN-RANK METRICS AND THE ONSET OF SUSTAINED RANK ONE. THE TWO SECOND-STAGE CONDITIONS MARKED $*$ USE PENALIZED MEAN RANK; THE REMAINING EIGHT USE GE OVER THE CANDIDATE SPACE DEFINED FOR EACH TARGET. STABLE N IS THE SMALLEST EVALUATED TRACE COUNT FROM WHICH THE METRIC REMAINS ONE AT EVERY SUBSEQUENT EVALUATION POINT THROUGH 5,000 TRACES.

Seed bits	Address	Recovery target	Mean rank (100)		Mean rank (500)		Stable N	
			f	s	f	s	f	s
128	FORS	W_5 via A'_5	20,353.033	20,628.800	1.900	1.667	1,000	1,000
128	FORS	W_9 via W_{16}	61.800	99.467	1.000	1.033	394	1,250
128	WOTS+	W_9 via W_{18}	1.000	1.000	1.000	1.000	96	86
256	FORS	Joint W_5/W_{13} (stage 1)	472.100	213.767	1.000	1.000	385	376
256	FORS	Joint W_5/W_{13} (stage 2)*	244,670.800	192,244.867	1.000	1.000	245	270

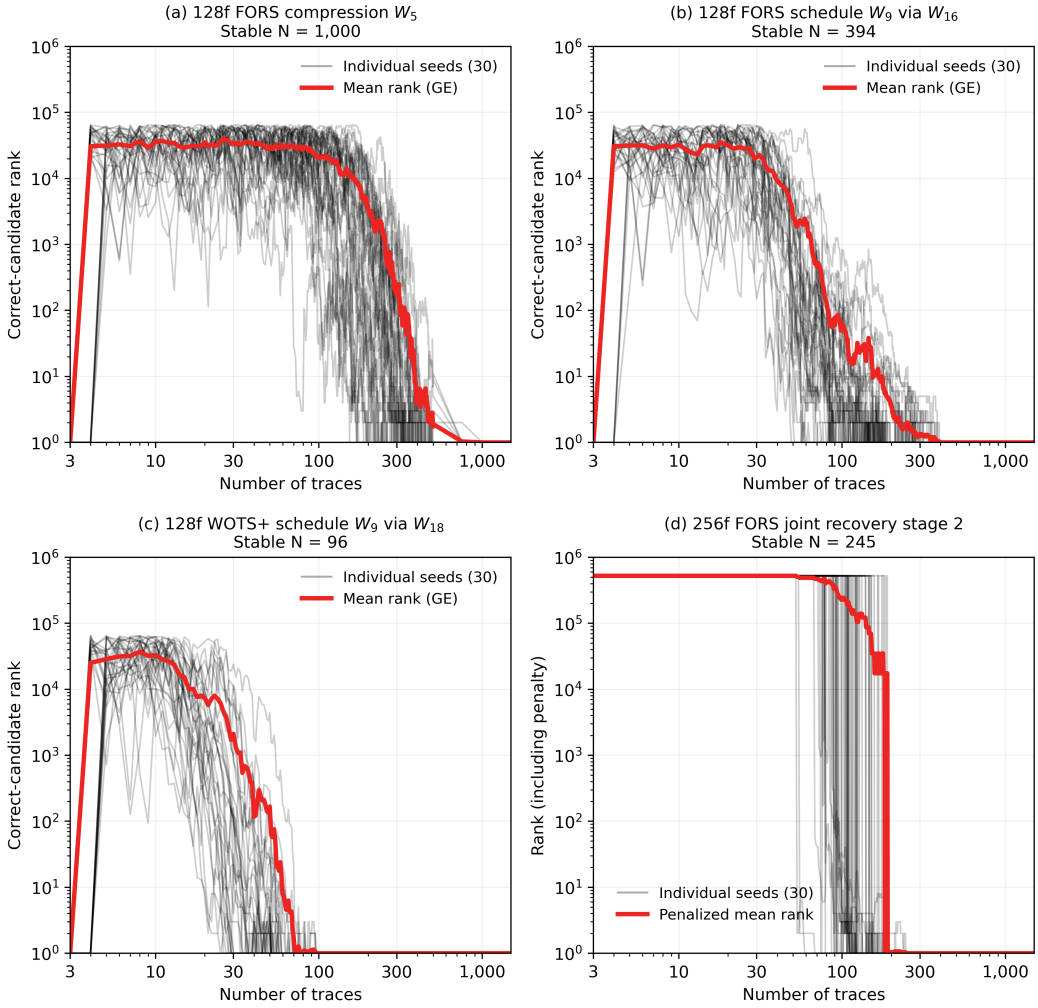


Fig. 5. Representative GE and penalized-mean-rank curves. Gray lines show individual values for 30 seeds, and red lines show their arithmetic mean. Panels (a)–(c) show correct-candidate ranks and GE. Panel (d) includes the penalty 524,289 when the correct class is eliminated. Both axes are logarithmic, with the common displayed range of 3–1,500 traces. Stable N is computed from all subsequent evaluation points through 5,000 traces, as defined in (18).

VII. DISCUSSION

A. Countermeasures and PRF-Call Shuffling

Appropriate masking can mitigate first-order secret-dependent leakage and has been applied to an SLH-DSA accelerator's Keccak/SHAKE path [14]. Masking SHA-256 software requires protecting both compression rounds and the message schedule, with additional computation, randomness, and implementation complexity. We do not

TABLE IV
COMPLETE-SEED RECOVERY USING PUBLIC ADDRESSES DERIVED FROM ONE SIGNATURE. EACH CONDITION USES THREE SEEDS AND THREE INDEPENDENTLY ANALYZED SIGNATURES PER SEED. SUCCESS REQUIRES A FULL SK.SEED MATCH USING ONLY THAT SIGNING PATH'S TRACES OF THE SPECIFIED PRF TYPE AND CHAINING ACTUAL RECOVERED VALUES. A DASH DENOTES AN UNEVALUATED MESSAGE-SCHEDULE-ONLY CONDITION, NOT 0%. COMP. AND SCHED. DENOTE COMPRESSION-ROUND AND MESSAGE-SCHEDULE CPA; 0/9 AND 9/9 CORRESPOND TO 0% AND 100%, RESPECTIVELY.

Parameter	PRF type	Traces per signature	Comp.	Sched.
128f	FORS	2,112	0/9	0/9
128f	WOTS+	6,160	9/9	9/9
128s	FORS	57,344	0/9	0/9
128s	WOTS+	125,440	9/9	9/9
192f	FORS	8,448	0/9	0/9
192f	WOTS+	8,976	9/9	—
192s	FORS	278,528	0/9	0/9
192s	WOTS+	182,784	9/9	—
256f	FORS	17,920	0/9	0/9
256f	WOTS+	18,224	9/9	—
256s	FORS	360,448	0/9	0/9
256s	WOTS+	137,216	9/9	—

TABLE V
COMPLETE-SEED RECOVERY BY COMPRESSION-ROUND CPA ON KEY-GENERATION PATHS.

Parameter	Traces per key generation	Recovery rate (%) and cases
128f	280	0 (0/3)
128s	17,920	100 (3/3)
192f	408	0 (0/3)
192s	26,112	100 (3/3)
256f	1,072	66.7 (2/3)
256s	17,152	100 (3/3)

measure these costs or claim an advantage over shuffling. Hierarchical derivation of individual secret values can reduce direct seed reuse [15]. Although compatible with standard verification, this approach changes the standard key-generation and signing procedures, so standards compliance must also be considered.

Shuffling randomizes independent operations to increase uncertainty about leakage timing [26]. Its protection depends on permutation construction and implementation leakage; combinations with masking likewise require evaluating cost and protection [26], [27]. Saarinen discusses randomizing FORS-tree processing and traversal orders during SLH-DSA signing [14], but does not report a concrete shuffling implementation or evaluate its protection and cost.

We theoretically examine four scopes for shuffling *initial PRF calls*, which generate FORS secret values or WOTS+ chain starting values. The call set follows the FIPS 205 FORS and XMSS signing pseudocode; Section VII-B1 derives its size [1]. Shuffling changes only the call order, preserving hash-internal operations and address–output correspondence. Thus, the call set follows the standard pseudocode, although the execution order changes.

An *XMSS group* is one height-h' XMSS tree at one hypertree layer for a signature. Authentication-path sibling subtrees are not separate groups. The four scopes are:

Local. Shuffle calls within each FORS tree and chain-start generation within each WOTS+ key. Preserve the reference order between trees and between WOTS+ keys.

Subtree. Shuffle all FORS initial PRF calls as one group, and all WOTS+ initial PRF calls within each XMSS group together. Process FORS first, then XMSS groups in their original layer order.

Hierarchical. Shuffle the k FORS groups and d XMSS groups together and shuffle calls within each group. This is the combined construction analyzed here. Each group's calls remain consecutive; retain their outputs until subsequent computation has its required inputs.

Global. Shuffle all initial FORS and WOTS+ PRF calls for one signature as one group, allowing calls from different trees and keys to interleave.

Subsequent chain and tree computations must wait for their inputs. Each selected FORS secret value is generated once and used where required. The scopes reorder different units of the same call set; their listing order does not imply an ordering of practical protection.

B. Theoretical Evaluation of Shuffling

1) Analysis Model and PRF-Call Counts: We analyze the FIPS 205 signing call set, without measuring a protected implementation. Let L and T count leaves per FORS tree and XMSS group, respectively, and let $l = \mathrm{len}$ count chains per WOTS+ key. Let h be the total hypertree height. Retaining the earlier definitions of n, a, h', k, d, we have

$$L = 2^a, \qquad T = 2^{h'}, \qquad h' = \frac{h}{d}. \tag{19}$$

The FIPS 205 FORS signing procedure generates the selected secret value once and generates the remaining secret values in sibling subtrees used to construct the authentication path. The number of initial PRF calls per FORS tree is therefore [1]

$$1 + \sum_{u=0}^{a-1} 2^u = 2^a = L. \tag{20}$$

In an XMSS tree, the WOTS+ key used for signing and the WOTS+ public keys for the other leaves each require

l initial PRF calls, yielding Tl calls per XMSS group. The counts for one signature are

$$N_{\mathrm{F}} = kL, \qquad N_{\mathrm{W}} = dTl, \qquad N = N_{\mathrm{F}} + N_{\mathrm{W}}. \quad (21)$$

Only address-based PRF calls processing `SK.seed` are counted. Key generation, $\mathrm{PRF}_{\mathrm{msg}}$, message hashing, and subsequent WOTS+ chain and tree hashes are excluded. We assume no cache that reuses values computed for earlier signatures.

2) Number of Possible PRF-Call Orders: Assume that individual calls are distinguishable and that permutations are selected uniformly within the permitted scope. The numbers of possible orders are

$$\begin{aligned} P_{\mathrm{Local}} &= (L!)^k (l!)^{dT}, \\ P_{\mathrm{Subtree}} &= (kL)!\big((Tl)!\big)^d, \\ P_{\mathrm{Hierarchical}} &= (k+d)!(L!)^k\big((Tl)!\big)^d, \\ P_{\mathrm{Global}} &= N!. \end{aligned} \quad (22)$$

Under a uniform within-group permutation, a particular call in a group of size m occupies each position with probability $1/m$. Hierarchical shuffling permutes k FORS groups of size L and d XMSS groups of size Tl. Since these group sizes may differ, uniformly permuting the groups does not place a particular PRF call uniformly over all positions in the complete call sequence.

Equation (22) counts possible orders; it does not measure the uncertainty supplied by an actual random-number generator or the work required by an attack. Whether an observation reveals group boundaries or allows public addresses to be identified again requires separate evaluation.

3) Number of Random Selections: We use Fisher–Yates permutation generation, in which the final remaining position is determined automatically [28]. A group of size m requires $m-1$ index selections. Define the randomness count D as the number of in-range indices actually accepted:

$$\begin{aligned} D_{\mathrm{Local}} &= k(L-1) + dT(l-1) \\ &= N - (k+dT), \\ D_{\mathrm{Subtree}} &= (kL-1) + d(Tl-1) \\ &= N - (d+1), \\ D_{\mathrm{Hierarchical}} &= (k+d-1) + k(L-1) \\ &\quad + d(Tl-1) = N-1, \\ D_{\mathrm{Global}} &= N-1. \end{aligned} \quad (23)$$

Hierarchical and Global have equal selection counts but different index ranges and possible permutations. Equal D does not imply equal random-bit consumption, execution time, or protection. Rejection sampling may require more random draws than D. Table VI excludes key-generation and signing randomness and generator initialization. Index swaps, address management, and output storage also contribute to permutation-generation overhead, which D alone does not measure.

TABLE VI
NUMBER OF ACCEPTED RANDOM INDEX SELECTIONS FOR EACH SHUFFLING SCOPE.

Parameter	PRF calls N	Local	Subtree	Hierarchical	Global
128f	8,272	8,063	8,249	8,271	8,271
128s	182,784	179,186	182,776	182,783	182,783
192f	17,424	17,215	17,401	17,423	17,423
192s	461,312	457,711	461,304	461,311	461,311
256f	36,144	35,837	36,126	36,143	36,143
256s	497,664	495,594	497,655	497,663	497,663

Table VI substitutes the six SHA-2-based SLH-DSA parameter sets into (23).

4) PRF-Output Buffer Size: A PRF output is the n-byte address-based PRF result, excluding derived WOTS+ endpoints and intermediate tree nodes. Our basic model buffers a group's complete PRF outputs before subsequent computation. Ready groups complete that computation and release their PRF storage before the next group's outputs are generated; XMSS groups awaiting inputs retain all their PRF outputs. Tree roots, signature fragments, and chain or tree intermediates stored outside groups are excluded.

Let $U(t)$ denote the number of PRF-output slots simultaneously allocated at time t. The peak buffer size is

$$M_{\mathrm{buf}} = n \max_t U(t). \quad (24)$$

Under this model, Local and Subtree require

$$M_{\mathrm{Local}} = n \max(L, l) \quad (25)$$

and

$$M_{\mathrm{Subtree}} = n \max(kL, Tl), \quad (26)$$

respectively.

With Hierarchical shuffling, upper XMSS groups may produce initial PRF outputs before the lower roots required for subsequent computation become available. If every XMSS group is waiting, at most dTl outputs are retained, and processing one FORS tree adds at most L outputs. Processing ready groups and releasing their PRF storage before generating the next group's outputs gives

$$M_{\mathrm{Hierarchical}} \le n(dTl + L). \quad (27)$$

This bound excludes tree roots, signature fragments, and other intermediate values stored outside the groups.

The basic Global model generates all initial PRF outputs before beginning subsequent computation, giving

$$M_{\mathrm{Global}} = nN. \quad (28)$$

TABLE VII
PRF-OUTPUT BUFFER SIZES IN BYTES UNDER THE SPECIFIED STORAGE MODEL. HIERARCHICAL VALUES ARE UPPER BOUNDS.

Parameter	Local	Subtree	Hierarchical (upper bound)	Global
128f	1,024	33,792	99,584	132,352
128s	65,536	917,504	2,072,576	2,924,544
192f	6,144	202,752	221,568	418,176
192s	393,216	6,684,672	4,780,032	11,071,488
256f	16,384	573,440	599,552	1,156,608
256s	524,288	11,534,336	4,915,200	15,925,248

Table VII gives the values from (25)–(28). The Hierarchical entries are upper bounds under the stated processing rules, not average usage. The table does not represent total RAM usage, measured increases over an unprotected implementation, or minimum memory requirements for every possible implementation.

Permutation arrays, call indices, random-number-generator state, stack usage, and the output signature buffer must be accounted for separately. For example, an array storing a b-byte index for each of the N calls requires bN bytes, but this array representation is not mandatory for every shuffling implementation. Placement in flash or RAM must also be distinguished in an actual implementation.

C. Limitations and Future Work

Conditional random-address success rates, representative rank metrics, and full-seed signature/key-generation success rates have different targets and assumptions. The signature experiment uses three seeds and three signatures per seed for each parameter set. Signatures sharing a seed do not represent different keys, and the observations do not guarantee success for all keys and messages.

Signature-path data preserve signature-derived addresses and call order, but record individual PRF executions rather than continuous signing. Identifying calls in full-signature traces and matching them to addresses remain unvalidated. Each recovery uses one signature; we do not pool signatures to estimate a minimum required count. Candidate scores cover each PRF trace's full recorded interval, so associating correlation peaks with actual instructions and intermediate values also requires further analysis.

Key-generation traces also record individual PRF executions; call identification in continuous key generation is unevaluated. Results from three seeds per parameter set cannot be generalized to success probabilities over all secret keys.

The shuffling analysis compares permutations, accepted random selections, and basic PRF storage for four scopes over the FIPS 205 signing call set. No protected firmware or corresponding power traces are evaluated, so the results establish neither device-specific measured overhead nor CPA protection. Addresses remain public, and reordering calls does not remove intra-PRF leakage.

Schupp et al. recently used leakage during Fisher–Yates permutation generation to reconstruct the execution order of a shuffled ML-DSA NTT [29]. This does not establish the same attack against SLH-DSA, but motivates a separate assessment of whether shuffled call orders can be identified again.

The call-level approach preserves input–output correspondence and could be examined for SHAKE, whose cost and protection are not evaluated here. Future work should account for computational dependencies and storage in an implementation, compare continuous full-signature traces before and after protection, and assess whether calls and public addresses can be identified again.

VIII. CONCLUSION

We presented compression-round and message-schedule CPA for the SHA-2-based SLH-DSA address PRF on an STM32F303 ARM Cortex-M4, accounting for parameter-dependent block layouts. With correct preceding values supplied, all evaluated compression-round stages achieved 100% conditional success; some 256-bit schedule stages were less reliable.

Passing actual recovered values between stages, WOTS+ compression-round CPA recovered `SK.seed` in all nine signature-path cases per parameter set. WOTS+ schedule CPA succeeded in all nine cases for each 128-bit set; neither route achieved full recovery for FORS.

Key-generation WOTS+ compression-round CPA succeeded in 11 of 18 cases, showing seed exposure during both signing and key generation in the evaluated setting.

The eight GE conditions and two penalized-mean-rank conditions for joint stage 2 all reached one at 5,000 traces under their specified rules. These ranks do not substitute for candidate-pruning or full-seed success rates.

Our four-scope shuffling analysis compares permutations, random selections, and PRF-output storage theoretically. Side-channel protection must consider both compression rounds and the message schedule. Full-execution call identification, broader key and measurement coverage, and measured shuffling cost and protection remain future work.

ACKNOWLEDGMENT

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP)–Information Technology Research Center (ITRC) 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%).

REFERENCES

[1] National Institute of Standards and Technology, "Stateless hash-based digital signature standard," National Institute of Standards and Technology, FIPS PUB 205, Aug. 2024.

[2] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The SPHINCS+ signature framework," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019, pp. 2129–2146.

[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, vol. 4867. Berlin, Germany: Springer, 2007, pp. 317–332.

[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 *Proceedings of the International Conference on Security and Cryptography (SECRYPT)*, 2013, pp. 230–241.

[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, vol. 13211. Cham, Switzerland: Springer, 2022, pp. 193–211.

[6] Y. Belenky *et al.*, "Carry-based differential power analysis (CDPA) and its application to attacking HMAC-SHA-2," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 3, pp. 1–29, 2023.

[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.

[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, vol. 15952. Cham, Switzerland: Springer, 2026, pp. 291–310.

[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, vol. 10808. Cham, Switzerland: Springer, 2018, pp. 1–20.

[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, vol. 5747. Berlin, Germany: Springer, 2009, pp. 66–80.

[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, vol. 12910. Cham, Switzerland: Springer, 2021, pp. 31–52.

[12] M. Lecomte, J. Maillard, A. Moran, G. Renault, and B. Smith, "Soft analytical side-channel attacks on SHA-2 and HMAC," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2026, no. 3, pp. 1205–1227, 2026.

[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, vol. 10815. Cham, Switzerland: Springer, 2018, pp. 168–188.

[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, vol. 14920. Cham, Switzerland: Springer, 2024, pp. 276–304.

[15] S. Fluhrer, "Side channel resistant SPHINCS+," Cryptology ePrint Archive, Paper 2024/500, 2024. [Online]. Available: https://eprint.iacr.org/2024/500

[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.

[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. [Online]. Available: https://eprint.iacr.org/2026/2584

[19] V. Satheesh and D. Shanmugam, "Implementation vulnerability analysis: A case study on ChaCha of SPHINCS," in *Proceedings of the IEEE International Symposium on Smart Electronic Systems (iSES)*, 2020, pp. 97–102.

[20] A. Genêt, "On protecting SPHINCS+ against fault attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 2, pp. 80–114, 2023.

[21] T. Huang *et al.*, "A timing attack resistant lightweight post-quantum crypto-processor for SPHINCS+," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2024, pp. 1–5.

[22] National Institute of Standards and Technology, "Secure hash standard (SHS)," National Institute of Standards and Technology, FIPS PUB 180-4, Aug. 2015.

[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, vol. 3156. Berlin, Germany: Springer, 2004, pp. 16–29.

[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 *Proceedings of the IET Irish Signals and Systems Conference (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

[26] N. Veyrat-Charvillon, M. Medwed, S. Kerckhof, and F.-X. Standaert, "Shuffling against side-channel attacks: A comprehensive study with cautionary note," in *Advances in Cryptology (ASIACRYPT 2012)*, ser. Lecture Notes in Computer Science, vol. 7658. Berlin, Germany: Springer, 2012, pp. 740–757.

[27] M. Azouaoui, O. Bronchain, V. Grosso, K. Papagiannopoulos, and F.-X. Standaert, "Bitslice masking and improved shuffling: How and when to mix them in software?" *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 2, pp. 140–165, 2022.

[28] R. Durstenfeld, "Algorithm 235: Random permutation," *Communications of the ACM*, vol. 7, no. 7, p. 420, Jul. 1964.

[29] J. Schupp, A. Baumeister, A. Wachter-Zeh, and G. Sigl, "Unshuffling Fisher–Yates: Practical recovery of length-128 permutations and application to ML-DSA," Cryptology ePrint Archive, Paper 2026/2658, 2026. [Online]. Available: https://eprint.iacr.org/2026/2658