Journal of Cryptologic Research, 2025, X(X): 1–21
DOI: 10.13868/j.cnki.jcr.000XXX
© The Editorial Office of *Journal of Cryptologic Research*
E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-82789618

Efficient Key Recovery via Correlation Power Analysis on Scloud$^+$

BAI Hang-Yu[1], HUANG Fan[1], DUAN Xiao-Lin[1], HU Hong-Gang[1,2]

1. School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230026, China

2. Hefei National Laboratory, Hefei 230088, China

Corresponding author: HU Hong-Gang, E-mail: hghu2005@ustc.edu.cn

Abstract: Scloud$^+$ is a next-generation post-quantum key encapsulation mechanism that combines unstructured-LWE and a ternary key encoding technique to achieve efficient lattice cryptographic operations while eliminating traditional ring structure constraints. Despite its rigorously formalized theoretical security, its practical deployment faces side-channel threats, notably Correlation Power Analysis (CPA) attacks. This paper systematically investigates the physical security of its core ciphertext-key matrix multiplication module by proposing a CPA framework that integrates instruction-level timing analysis. A SoST (Sum of Squared T-values) model, inspired by multi-group Welch's t-test, is used to analyze the Hamming weight leakage during ciphertext loading. At the same time, dynamic sampling windows, combined with processor pipeline modeling, are employed to pinpoint critical leakage intervals. Exploiting the characteristics of ternary keys, an iterative recovery strategy is devised: following a predefined scan order, the candidate set $\{-1, 0, 1\}$ and partial intermediate sums are used to construct a Hamming weight model for hypothesized leakage vectors. Pearson correlation analysis and trace-count stabilization are applied within each dynamic sampling window to determine the optimal estimate for each key element. Experiments targeting 4800 key elements, illustrated through a detailed analysis of the first 32 elements, demonstrate high recovery accuracy with no more than 15 traces per element, indicating high efficiency and stability that can extend to the full key reconstruction. To thwart such CPA attacks, we have further designed and implemented a first-order arithmetic masking countermeasure that splits the original ternary secret key into two subkeys, thereby expanding the attacker's key hypothesis space and significantly enhancing side-channel resilience. Our results demonstrate that Scloud$^+$ remains vulnerable to side-channel exploits at the implementation level, highlighting the urgent need to integrate appropriate protections into its standardization process.

Key words: side channel attack; correlation power analysis; post-quantum cryptography; Scloud$^+$

Cite: ZUO Z Y, ZUO Z E, ZUO Z S. Efficient key recovery via correlation power analysis on scloud$^+$ [J]. Journal of Cryptologic Research, 2025, X(X): 1–21. [DOI: 10.13868/j.cnki.jcr.000XXX]

1 Introduction

Public key cryptography is increasingly threatened by the rapid development of quantum computing, particularly due to Shor' s algorithm [1], which efficiently breaks widely used schemes based on integer factorization and discrete logarithms. In response, post-quantum cryptographic proposals have gained significant attention. Among them, Scloud$^+$ [2] has emerged as a promising lattice-based key

* Foundation: National Natural Science Foundation of China (62472397)
Received: 20XX-XX-XX Revised: 20XX-XX-XX

encapsulation mechanism (KEM), offering high efficiency while avoiding algebraic structures that may weaken security. Although its theoretical security is supported by formal proofs under the Learning With Errors (LWE) assumption, its practical resilience against physical attacks remains an open concern. In particular, side-channel attacks (SCA)—which exploit information leaked through power consumption, timing, or electromagnetic emissions—pose substantial risks to real-world deployments [3]. Correlation Power Analysis (CPA), as one of the most effective SCA techniques, represents a critical threat vector that warrants further investigation.

The strength of Correlation Power Analysis lies in its ability to exploit linear correlations between intermediate cryptographic computations and observed power consumption patterns [4]. By hypothesizing intermediate values based on key guesses and computing their statistical correlation with recorded power traces, CPA achieves high attack efficiency with minimal implementation overhead. Compared to Differential Power Analysis (DPA) [5], CPA offers superior effectiveness while maintaining comparable methodological simplicity.

Recently, CPA has made substantial progress in targeting lattice-based cryptographic schemes. For instance, the study in [6] applied Pearson correlation analysis to power traces from NTRU implementations on RFID cards, achieving key recovery by exploiting vulnerabilities in schoolbook polynomial multiplication. Espitau et al. (2017) further demonstrated the feasibility of electromagnetic CPA attacks against the BLISS signature scheme by targeting its polynomial multiplication operations [7]. Aysu et al. (2018) introduced the concept of horizontal CPA, which leverages intra-trace correlations across multiple intermediate values to recover keys in NewHope and Frodo key exchange protocols, attaining a success rate of 99% [8]. Huang et al. (2020) [9] proposed a multi-target CPA framework for NTRU Prime, using product-scanning multiplication [10] and Hamming weight-based leakage models to iteratively reconstruct nonzero key bits. Concurrently, Fournaris et al. (2020) presented a three-phase CPA methodology tailored for the Dilithium signature scheme, involving variable randomization, hypothesis construction, and Pearson correlation analysis to extract secret key material from the computation of $Z = Y + C \cdot S_1$ [11]. More recently, a two-step CPA-based lattice attack against CRYSTALS-Kyber has been proposed, where partial secret keys are first recovered from NTT-based polynomial multiplications and subsequently extended via lattice reduction, further demonstrating the scalability and potency of CPA techniques in post-quantum settings [12].

Motivated by the lack of side-channel analysis on Scloud$^+$ and the critical role of correlation power analysis in evaluating the practical security of post-quantum lattice-based cryptography, this work investigates the physical leakage risks associated with its core operations. While Scloud$^+$ offers strong theoretical security guarantees under the Learning With Errors (LWE) problem without structured algebraic assumptions, its resistance against real-world implementation attacks remains unexplored. We propose a novel approach based on correlation power analysis, enabling efficient key recovery using minimal traces. Our findings demonstrate that despite its theoretical robustness, Scloud$^+$ exhibits practical vulnerabilities that can be exploited via side-channel attacks. To mitigate such threats, we design and implement a first-order arithmetic masking countermeasure that splits the original ternary secret key into two uniformly distributed subkeys. This strategy significantly increases the key hypothesis space, thereby enhancing resistance against the attack and offering a practical defense mechanism for secure deployments.

1.1 Our Contributions

The main contributions of this paper are summarized as follows.

- We are the first to present the first side-channel attack targeting Scloud$^+$, revealing its vulnerability to correlation power analysis. Our findings demonstrate that despite its post-quantum theoretical security, Scloud$^+$ suffers from practical implementation-level leakage.

- We propose a novel attack framework that integrates dynamic sampling window selection with instruction-level timing characterization to precisely locate the store operations involving intermediate values. This precision enables full recovery of the first 32 secret key elements using fewer than 15 traces per element. We further analyze the feasibility of extending this method to iteratively recover all 4800 secret key elements, indicating the scalability and practicality of our approach.

- We also propose the countermeasure against the attack. We design and evaluate a first-order arithmetic masking countermeasure that splits the ternary secret matrix into two full-range additive shares, thereby increasing the attacker' s search space and effectively thwarting CPA-based leakage.

1.2 Organization

The remainder of this paper is organized as follows. Section 2 introduces the preliminary concepts, including the notation and algorithmic framework of the target system, Scloud$^+$-128, and provides a concise overview of CPA. Section 3 presents the core methodology for mounting CPA attacks against Scloud$^+$, detailing the underlying principles and attack strategy. Section 4 validates the proposed approach through experimental results, demonstrating its effectiveness and scalability in practical settings. Section 5 describes our first-order arithmetic masking countermeasure designed to mitigate the identified leakage. Finally, Section 6 concludes the paper and outlines directions for future research.

2 Preliminaries

In this section, we first present the notations and parameters for Scloud$^+$-128. Then, we review the Scloud$^+$.PKE and Scloud$^+$.KEM algorithms. Finally, we introduce the basic principle of CPA.

2.1 Notations

The following conventions and variables are defined for Scloud$^+$-128:

- **Vector and Matrix Notation:** Bold lowercase symbols (e.g., $\mathbf{v}$) are used to represent vectors, while bold uppercase symbols (e.g., $\mathbf{A}$) denote matrices throughout this work.

- **Probabilistic Sampling:** Selection from a distribution χ is expressed as $x \leftarrow \chi$. Uniform sampling over a finite set S is denoted by $U(S)$.

- **Modular Arithmetic Parameters:** Prime powers $q > q_1, q_2$, where $q = 2^{12}$, $q_1 = 2^9$, and $q_2 = 2^7$ for Scloud$^+$-128.

- **Matrix Dimensions:** Core matrix dimensions $(m, n) = (600, 600)$, with reduced dimensions $(\bar{m}, \bar{n}) = (8, 8)$.

- **Secret Key Sparsity Constraints:** Weight parameters $(h_1, h_2) = (150, 150)$, enforcing ternary secret distributions.

- **Error Distribution Bounds:** Noise parameters $(\eta_1, \eta_2) = (7, 7)$, governing error magnitudes in lattice operations.

- **Message and Shared Secret Lengths:** Plaintext length $l_m = 128$ bits and shared secret length $l_{ss} = 128$ bits.

- **Labeling Granularity:** Constants $\mu = 64$ and $\tau = 3$ used in lattice-based encoding operations.

2.2 Scloud$^+$.PKE and Scloud$^+$.KEM

Algorithm 1 provides an overview of the Scloud$^+$.PKE scheme, which consists of three primary procedures: KeyGen, Enc, and Dec. Within this framework, the key generation function, Scloud$^+$.PKE.KeyGen, takes a uniformly sampled 256-bit random seed $\alpha \leftarrow U(\{0,1\}^{256})$ as input and outputs a corresponding public-private key pair (pk, sk). The encryption function, Scloud$^+$.PKE.Enc, then employs the public key pk, a plaintext message m, and a randomness input r to produce the ciphertext C. In contrast, decryption is handled by Scloud$^+$.PKE.Dec, which utilizes the private key sk and the ciphertext C to reconstruct the original message m. This encryption mechanism satisfies the IND-CPA (indistinguishability under chosen plaintext attack) security criterion.

Algorithm 1: Scloud$^+$.PKE Algorithm

$(pk, sk) = \text{Scloud}^+\text{.PKE.KeyGen}(\alpha)$
  $(\text{seed}_A, r_1, r_2) = F(\alpha) \in \{0,1\}^{128} \times \{0,1\}^{256} \times \{0,1\}^{256}$
  $A = \text{gen}(\text{seed}_A) \in \mathbb{Z}_q^{m \times n}$
  $S = \Psi(r_1, (n, \bar{n}), h_1) \in \mathbb{Z}^{n \times \bar{n}}$
  $E = \text{CenBinom}(r_2, (m, \bar{n}), \eta_1) \in \mathbb{Z}^{m \times \bar{n}}$
  $B = A \cdot S + E \in \mathbb{Z}_q^{m \times \bar{n}}$
  return $pk = (B, \text{seed}_A), sk = S$
$(C) = \text{Scloud}^+\text{.PKE.Enc}(pk, m, r)$
  $A = \text{gen}(\text{seed}_A)$
  $(r_1', r_2') = F(r) \in \{0,1\}^{256 \times 2}$
  $S' = \Phi(r_1', (\bar{m}, m), h_2) \in \mathbb{Z}^{\bar{m} \times m}$
  $E' = (E_1, E_2) = \text{CenBinom}(r_2', (\bar{m}, n + \bar{n}), \eta_2)$, where $E_1 \in \mathbb{Z}^{\bar{m} \times n}$, $E_2 \in \mathbb{Z}^{\bar{m} \times \bar{n}}$
  $M = \text{MsgEnc}(m) \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$
  $C_1 = S' \cdot A + E_1,\ C_2 = S' \cdot B + E_2 + M$
  $\bar{C}_1 = \left\lfloor \frac{q_1}{q} \cdot C_1 \right\rceil,\ \bar{C}_2 = \left\lfloor \frac{q_2}{q} \cdot C_2 \right\rceil_{\text{odd}}$
  return $C = (\bar{C}_1, \bar{C}_2)$
$(m) = \text{Scloud}^+\text{.PKE.Dec}((sk, C))$
  $C_1' = \frac{q}{q_1} \cdot \bar{C}_1,\ C_2' = \frac{q}{q_2} \cdot \bar{C}_2$
  $D = C_2' - C_1' S \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$
  return $m = \text{MsgDec}(D) \in \{0,1\}^{l_m}$

Similarly, the Scloud$^+$.KEM protocol is delineated in Algorithm 2, comprising three fundamental procedures: KeyGen, Encaps, and Decaps. Within this framework, the key generation function, Scloud$^+$.KEM.KeyGen, takes as input a random seed α and a uniformly sampled random vector $z \leftarrow U(\{0,1\}^{256})$, and outputs a corresponding public-private key pair (pk, sk). The encapsulation operation, Scloud$^+$.KEM.Encaps, employs the public key pk together with a message m to generate a ciphertext C and a shared secret ss. In contrast, the decapsulation process, Scloud$^+$.KEM.Decaps, takes the secret key sk and ciphertext C as inputs to retrieve the shared secret ss. This key encapsulation scheme conforms to the IND-CCA2 security model, ensuring robustness against chosen ciphertext attacks. To conduct a CPA-based side-channel evaluation, our implementation specifically isolates the critical multiplication operation $C_1' \cdot S$ from the decapsulation phase of Scloud$^+$.KEM.Decaps, enabling precise analysis of potential leakage.

Algorithm 2: Scloud+.KEM Algorithm

$(pk, sk) = \text{Scloud}^+\text{.KEM.KeyGen}(\alpha, z)$
  $(pk, sk') = \text{Scloud}^+\text{.PKE.KeyGen}(\alpha)$
  $hpk = H(pk) \in \{0,1\}^{256}$
  $sk = (sk', pk, hpk, z)$
  return (pk, sk)
$(C, ss) = \text{Scloud}^+\text{.KEM.Encaps}(pk, m)$
  $(r, k) = G(m \parallel H(pk)) \in \{0,1\}^{256\times 2}$
  $C = \text{Scloud}^+\text{.PKE.Enc}(pk, m, r)$
  $ss = K(k \parallel C) \in \{0,1\}^{l_{ss}}$
  return (C, ss)
$(ss) = \text{Scloud}^+\text{.KEM.Decaps}(sk, C)$
  $m' = \text{Scloud}^+\text{.PKE.Dec}(sk', C)$
  $(r', k') = G(m' \parallel hpk)$
  $C' = \text{Scloud}^+\text{.PKE.Enc}(pk, m', r')$
  if $C = C'$ **then**
    return $ss = K(k' \parallel C)$
  end
  else
    return $ss = K(z \parallel C)$
  end

2.3 Test Vector Leakage Assessment

Test Vector Leakage Assessment (TVLA) [13], introduced by Goodwill et al., serves as a robust statistical technique for detecting potential side-channel leakages in cryptographic implementations. The fundamental principle of TVLA involves performing a comparative analysis between two distinct sets of side-channel measurements to determine whether statistically significant differences exist, thereby indicating the presence of information leakage.

To facilitate this comparison, TVLA employs Welch' s t-test to determine whether a statistically significant difference exists between the means of two independent datasets. The test operates under the null hypothesis, which posits that both sets originate from populations with equal means. Rejection of this hypothesis suggests the presence of a substantial disparity, thereby indicating potential information leakage.

The TVLA score between two datasets T_0 and T_1 is computed as follows:

$$\text{TVLA} = \frac{\mu_0 - \mu_1}{\sqrt{\frac{\sigma_0^2}{n_0} + \frac{\sigma_1^2}{n_1}}} \tag{1}$$

where μ_i, σ_i, and n_i (for $i \in \{0,1\}$) denote the mean, standard deviation, and sample size of each dataset, respectively. A TVLA score exceeding an absolute threshold of 4.5 indicates a statistically significant deviation at the 99.9999% confidence level [13], thereby rejecting the null hypothesis.

In this study, the TVLA methodology is adopted as a preliminary step to identify Points of Interest (POIs) [14] prior to conducting CPA attacks. A refined statistical indicator, the **Sum of Squared T-values (SoST)**, is utilized to improve the sensitivity and reliability of leakage detection. Built upon the classical Welch' s t-test used in traditional TVLA, the SoST metric enhances its effectiveness by aggregating pairwise t-statistics across all distinct data-dependent groups. This cumulative evaluation facilitates a more comprehensive characterization of side-channel leakage. Formally, let $\mu_w \in \mathbb{R}^L$ and $\sigma_w^2 \in \mathbb{R}^L$ denote the mean and variance vectors of traces associated with group w, where L represents

the trace length. The SoST value at each sample point is computed as:

$$\mathrm{SoST} = \sum_{q \leq p} \frac{(\mu_p - \mu_q)^2}{\frac{1}{N}(\sigma_p^2 + \sigma_q^2)} \tag{2}$$

where N denotes the total number of groups under consideration.

2.4 Correlation Power Analysis: Hamming Weight Power Model

In CPA, the central objective is to construct an accurate power consumption model that characterizes the behavior of the target device during cryptographic operations. The crux of the attack lies in the identification of a statistical correlation between the hypothetical power consumption, as predicted by the model, and the actual power traces obtained from the device. A strong correlation suggests that the hypothetical key used to generate the prediction is likely correct. With a sufficiently accurate model, acquiring a large set of power traces enables the attacker to isolate the key candidate that yields the highest correlation, thereby facilitating successful key recovery.

A widely adopted model in this context is the Hamming Weight (HW) Power Model. The Hamming weight of a binary value is defined as the total number of bits set to '1'. For example, the binary string 11110010 has a Hamming weight of 5. The fundamental premise of this model in power analysis attacks is that the number of bits set to 0 or 1 of an output is correlated with the power consumption of a device. Therefore, the Hamming weight provides a convenient and effective metric for approximating dynamic power consumption during data processing. In CPA-based attacks, it is also essential to have prior knowledge of either the ciphertext or the secret key. This prerequisite is rooted in the fact that the Hamming Weight (HW) Power Model estimates the number of bits set to '1' in intermediate computational values, particularly those involving known inputs such as the ciphertext or secret key, during the execution of cryptographic operations [15].

In practice, the predicted Hamming weights of intermediate cryptographic values are computed under different key hypotheses and are statistically compared against the actual voltage measurements collected during execution. This comparison is typically performed using the Pearson correlation coefficient as defined in Equation (2).

$$\rho(W, P) = \frac{\mathrm{Cov}(W, P)}{\sqrt{\mathrm{Var}(W) \cdot \mathrm{Var}(P)}} \tag{3}$$

In this formulation, W denotes the set of measured power consumption values, while P corresponds to the estimated power values derived from the Hamming weights. $\mathrm{Cov}(\cdot)$ represents the covariance and $\mathrm{Var}(\cdot)$ denotes the variance function. By applying Pearson's correlation, CPA determines whether a statistically significant relationship exists between the modeled and actual power profiles, thereby indicating potential leakage that aligns with the correct key hypothesis.

It is hypothesized that the ciphertext will be reconstructed while the secret key remains hidden. The correlation coefficient ρ is then computed between the actual power consumption measurements and the Hamming weight estimates of the intermediate results D, which are produced during the matrix multiplication operation $C_1' \cdot S$ in the decryption process.

3 Correlation Power Analysis on Scloud$^+$ Implementations

3.1 On Side Channel Leakage of the Scloud$^+$ Secret Key

Matrix multiplication is a cornerstone in lattice-based cryptographic constructions due to its frequent involvement of public and secret-dependent operands, rendering it highly vulnerable to SCA [16]. The observable leakages in hardware implementations stem from a combination of concurrent physical

phenomena. For instance, memory address decoding generates distinct power consumption patterns that vary depending on the accessed memory location. Similarly, transitions across the data bus reveal measurable characteristics linked to the Hamming weight and the specific bit composition of the transmitted values. Register-level operations further contribute to leakage through data-dependent power variations. In the Scloud$^+$ architecture, the Matrix Multiplier (MatM) module is central to executing key operations, including $A \cdot S$, $S' \cdot A$, $S' \cdot B$, and $C'_1 \cdot S$ [17]. Among these, the computation of $C'_1 \cdot S$ within the KEM.Decaps procedure is particularly noteworthy, as it involves a publicly known matrix C'_1 and a secret matrix S. This configuration allows an attacker to exploit CPA by monitoring the storage of intermediate results within processor registers.

3.2 Leakage Localization via SoST Analysis

The goal of the localization phase is to identify the precise timing of each $C'_1[i] \cdot S[i]$ operation within the recorded power traces. This process is inherently based on observable characteristics of the hardware during matrix multiplication. At the beginning of each iteration in the matrix multiplication loop, a candidate element from C'_1 is fetched from memory, generating distinguishable patterns in power consumption or electromagnetic emissions. These patterns are strongly correlated with the Hamming weight of the accessed data and are shaped by low-level hardware behaviors such as memory access and register operations. In essence, accurately pinpointing the load operation for each C'_1 element enables the determination of the starting sample point of the corresponding $C_i \cdot S_i$ multiplication. Consequently, the sampling window for CPA is defined as the interval between two successive load instructions.

The experimental design creates optimal conditions for leakage observation by maintaining strict control over the cryptographic state. To precisely localize every load operation for the candidate element of C'_1, all variables of C'_1 and S are set to zero except the candidate to isolate the target's contribution to the side-channel signal.

The experimental approach adopts a systematic grouping structure to thoroughly examine transitions in Hamming weight, a design choice motivated by considerations of hardware efficiency and the requirements of side-channel analysis. While Section 2.1 described a modulus of $q = 12$ for the ciphertext elements, the implementation adapts this to $q = 17$, in line with the 16-bit representation of the C'_1 elements. To achieve a uniform distribution across the full spectrum of Hamming weights, a dedicated `Sample` procedure is employed, wherein 10 distinct values are selected for each Hamming weight in the range $[0, 16]$. This ensures that each sampled input precisely conforms to its designated Hamming weight profile, thereby capturing a comprehensive range of data-dependent leakage behaviors.

To systematically assess side-channel vulnerabilities across this range, the SoST metric is applied across the 17 Hamming weight groups. Moreover, including multiple samples per weight group strengthens the statistical robustness of the computed SoST values. Algorithm 3 outlines this methodology, integrating physical effects with refined statistical processing to enable effective localization of leakage points within power traces.

Algorithm 3 Localization via SoST Analysis

Input: Index of C_1' $CIndex$
Output: SoST result within a file
$\mu \leftarrow [\mu_0, \ldots, \mu_{16}]$;
$\sigma^2 \leftarrow [\sigma_0^2, \ldots, \sigma_{16}^2]$;
$[S_0, \ldots, S_{n*\bar{n}-1}] \leftarrow 0$
$Sample$(C1Xs);
for $hw \leftarrow 0$ **to** 16 **do**
  for $k \leftarrow 0$ **to** 9 **do**
    $data[\,] \leftarrow 0$ except $data[CIndex] \leftarrow C1Xs[hw][k]$;
    $tr \leftarrow$ captureTrace($scope$, $target$, $data$, S);
    $trLoc[hw].append(tr[0])$
  end
  $\mu_{hw} \leftarrow$ average($trLoc[hw]$);
  $\sigma_{hw}^2 \leftarrow$ variance($trLoc[hw]$);
end
$SoST \leftarrow$ compute_SoST(μ, σ^2);
store SoST result for Localization;

3.3 Secret Inference Based on Correlation Power Analysis

Inspecting the Scloud$^+$ specifications, the secret key is denoted as $S \in \mathbb{Z}_q^{n \times \bar{n}}$ having ternary values $\{-1, 0, 1\}$, and C_1' is defined as $C_1' \in \mathbb{Z}_q^{\bar{m} \times n}$. The computation of $C_1' \cdot S$ in line 19 of Algorithm 2 introduces predictable side-channel leakage during matrix multiplication operations. As demonstrated in Algorithm 4, the nested loops for computing

$$D[i][j] = \sum_k C[i][k] \cdot S[j][k] \tag{4}$$

create distinct power consumption patterns when intermediate results are stored in registers. This enables CPA attacks by correlating hypothesized power models with measured traces.

Algorithm 4 Matrix multiplication $C_1' \cdot S$ for Capture

Input: Ciphertext $C = C_1' \in \mathbb{Z}_q^{\bar{m} \times n}$, Secret Key $S \in \mathbb{Z}_q^{n \times \bar{n}}$
Output: Intermediate result $D \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$
Trigger_High();
for $i \leftarrow 0$ **to** $\bar{m} - 1$ **do**
  for $j \leftarrow 0$ **to** $\bar{n} - 1$ **do**
    $D[i][j] \leftarrow 0$;
    for $k \leftarrow 0$ **to** $n - 1$ **do**
      $D[i][j] \leftarrow D[i][j] + C[i][k] \cdot S[j][k]$;
    end
  end
end
Trigger_Low();

The optimized CPA attack is outlined in Algorithm 5: (1) adaptive acquisition of power traces within dynamically selected windows Γ_j corresponding to the computation of $D[i]$; (2) iterative hypothesis generation and partial sum reconstruction tailored to ternary secret key elements; (3) correlation maximization over refined sampling intervals and candidate key values to identify the most likely secret components.

Algorithm 5 CPA Attack Process

Input: Ciphertext $C = C_1'$, secret key S, number of key elements n ($0 < n \leq 600$), dynamic sampling window Γ

Output: Recovered key $\hat{s}$

Phase 1: Power Trace Collection
Load input data: $\{\mathbf{C}_1^{(i)}\}_{i=1}^{1000}$ and fixed secret S;
for $i \leftarrow 0$ **to** 99 **do**
  Acquire power trace: $\tau_i \leftarrow \text{captureTrace}(scope, target, \mathbf{C}_1^{(i)}, S)$;
end
Construct trace matrix $\mathcal{T} \in \mathbb{R}^{100 \times L}$ where L=trace length;
Phase 2: Iterative Key Recovery
Initialize recovered key $\hat{\mathbf{s}} \leftarrow \varnothing$;
for element index $j \leftarrow 0$ **to** $n-1$ **do**
  $\Gamma_j \leftarrow [\gamma_{\text{start}}^{(j)}, \gamma_{\text{end}}^{(j)}]$;
  for trace count $N \leftarrow 2$ **to** 100 **do**
    for sample γ **in range** Γ_j **do**
      Extract power vector $T_\gamma \leftarrow \{\mathcal{T}[i,\gamma]\}_{i=0}^{N-1}$;
      Initialize max correlation $\rho_{\max} \leftarrow -\infty$;
      for candidate key $k \in [-1, 0, 1]$ **do**
        for trace $i \leftarrow 0$ **to** $N-1$ **do**
          Compute intermediate value: $v_k^{(i)} \leftarrow \sum_{m=0}^{j-1} C_1^{(i)}[m] \cdot \hat{s}_m + C_1^{(i)}[j] \cdot k$;
          Calculate Hamming weight: $h_k[i] \leftarrow HW(v_k^{(i)})$;
        end
        Compute Pearson correlation:$\rho(k,\gamma) \leftarrow \left| \frac{\mathbb{E}[(h_k - \mu_{h_k})(T_\gamma - \mu_{T_\gamma})]}{\sigma_{h_k} \sigma_{T_\gamma}} \right|$
        Phase 3: Maximization of the correlation
        if $\rho(k,\gamma) > \rho_{\max}$ **then**
          Update $\rho_{\max} \leftarrow \rho(k,\gamma)$;
          Record candidate $key \leftarrow k$;
        end
      end
    end
    Record candidate $\hat{s}_j^{(N)} \leftarrow key$;
  end
  Determine stable recovery:$\hat{s}_j \leftarrow \texttt{majority}(\{\hat{s}_j^{(N)}\}_{N=2}^{100})$
  Update recovered key: $\hat{\mathbf{s}} \leftarrow \hat{\mathbf{s}} \cup \{\hat{s}_j\}$;
end
return $\hat{\mathbf{s}}$;

3.3.1 Optimized Localization Strategy via Dynamic Sampling Window

The attack begins by capturing trace 100 times with varying C_1' to construct a power trace matrix $\mathbf{T} \in \mathbb{R}^{100 \times L}$, where L is the number of sampling points per execution. Each trace

$$\tau_i = [t_{i1}, t_{i2}, \ldots, t_{iL}]^\top \tag{5}$$

captures the ARM processor's power consumption during the $C_1' \cdot S$ computation. To enhance efficiency, a *dynamic sampling window*

$$\Gamma_j = [\gamma_{\text{start}}(j), \gamma_{\text{end}}(j)] \tag{6}$$

is adaptively selected for locating each process of $C_1'[j] \cdot S[j]$ based on clock synchronization and leakage profiling.

In practice, the configuration of the dynamic sampling window Γ_j can be further refined by

leveraging microarchitectural insights into the target device. When employing the Hamming Weight (HW) model to simulate power consumption during the storage of intermediate results $C_1'[j] \cdot S[j]$, it is essential to localize the corresponding store operation in the power trace precisely.

To this end, a bottom-up inspection of the compiled assembly reveals a consistent instruction pattern associated with each matrix multiplication iteration as shown below:

```
8000758: f831 cb02    ldrh.w ip, [r1], #2
800075c: f83e 9b02    ldrh.w r9, [lr], #2
8000760: f8b4 8000    ldrh.w r8, [r4]
8000764: 428e         cmp    r6, r1
8000766: fb1c 8c09    smlabb ip, ip, r9, r8
800076a: f8a4 c000    strh.w ip, [r4]
800076e: d1f3         bne.n 8000758
```

Assembly routine of matrix multiplication loop

This instruction sequence reflects a fused multiply-accumulate operation followed by a store to memory (STR). Notably, the subsequent instructions—conditional branch (BNE) and the load (LDR) of the next loop—form a recurring pattern that provides a reliable timing reference. The maximum SoST value is attributable to the LDR instructions, as demonstrated in Section 3.2. In addition, the sampling rate is set to 4 samples per clock cycle, and the clock cycle of the three instructions costs between 4 and 8 cycles. In such cases, the peak is typically observed to occur 16 to 32 samples after the store instruction that leaks the multiplication result. Consequently, the leakage associated with the target $S[j]$ is most prominent within a pre-peak interval of 16–32 samples before the local SoST maximum.

Hence, to maximize attack efficiency, the adaptive window $\Gamma_j = [\gamma_{\text{start}}(j), \gamma_{\text{end}}(j)]$ is enlarged to include this leakage-prone zone, ensuring that the correlation analysis captures the precise moment of side-channel information release. This principled localization strategy significantly improves the quality of hypothesis testing and boosts the overall success rate of key recovery.

3.3.2 Iterative Key Recovery with Partial Sums

The ternary secret elements $S[j]$ are recovered iteratively using a bit-wise progressive strategy. For each target position j, the attacker hypothesizes candidate values $s \in \{-1, 0, 1\}$ and computes partial intermediate sums:

$$v_s^{(i)} = \sum_{m<j} C_1^{(i)}[m] \cdot \hat{S}[m] + C_1^{(i)}[j] \cdot s, \tag{7}$$

where $\hat{S}$ denotes the previously recovered elements, and the ordering $m < j$ follows a predefined scan order (C is row-major and S is column-major). The sums $v_s^{(i)}$ are computed for each power trace i, where $0 \le i \le N-1$, and for each candidate s.

Each $v_s^{(i)}$ is then mapped to a hypothetical leakage value using a Hamming Weight (HW) model:

$$h_s[i] = HW(v_s^{(i)}) = \sum_{b=0}^{w-1} \left(\frac{v_s^{(i)}}{2^b} \bmod 2 \right), \tag{8}$$

where w denotes the bit-width of the $v_s^{(i)}$. All $3 \times N$ hypothesis leakage vectors h_s are computed in parallel for efficiency.

3.3.3 Adaptive Correlation Maximization with Trace-Count Stabilization

For each sampling point $\gamma \in \Gamma_j$, the Pearson correlation coefficient between the hypothetical leakage vector h_s and the measured power trace samples $\mathcal{T}_\gamma = T[:, \gamma]$ is computed as:

$$\rho(s, \gamma) = \left| \frac{\sum_{i=0}^{N-1} (h_s[i] - \mu_{h_s})(\mathcal{T}_\gamma[i] - \mu_{\mathcal{T}_\gamma})}{\sqrt{\sum_{i=0}^{N-1} (h_s[i] - \mu_{h_s})^2} \cdot \sqrt{\sum_{i=0}^{N-1} (\mathcal{T}_\gamma[i] - \mu_{\mathcal{T}_\gamma})^2}} \right|, \tag{9}$$

where μ_{h_s} and $\mu_{\mathcal{T}_\gamma}$ denote the empirical means of h_s and $\mathcal{T}_\gamma$ respectively. This produces $3 \times |\Gamma_j|$ correlation values per trace count N.

The candidate selection first performs joint maximization over all candidates and sampling points for each trace count $N \in \{2, \ldots, 100\}$:

$$\hat{S}[j]^{(N)} = \arg \max_{s \in \{-1,0,1\}} \max_{\gamma \in \Gamma_j} \rho(s, \gamma), \tag{10}$$

The final key estimate $\hat{S}[j]$ is then determined through trace-count stabilization analysis:

$$\hat{S}[j] = \texttt{mode}\left(\{\hat{S}[j]^{(N)}\}_{N=N_{\min}}^{100} \right), \quad N_{\min} = \min \left\{ N \mid \hat{S}[j]^{(k)} = \hat{S}[j]^{(N)} \ \forall k \geq N \right\} \tag{11}$$

where $\texttt{mode}(\cdot)$ returns the persistent value that appears for all $N \geq N_{\min}$, establishing the minimum trace count required for stable recovery.

The complete secret key vector $\hat{\mathbf{s}}$ is constructed by aggregating stabilized estimates across all positions:

$$\hat{\mathbf{s}} = \left(\hat{S}[0], \hat{S}[1], \ldots, \hat{S}[n-1] \right) \in \{-1, 0, 1\}^n \tag{12}$$

where each element $\hat{S}[j]$ satisfies the stabilization criterion defined in (11). This synthesis process guarantees that:

- $\hat{S}[j]$ remains constant for $N \geq N_{\min}^{(j)}$
- The combined vector $\hat{\mathbf{s}}$ achieves $\prod_{j=0}^{n-1} \mathbb{I}(\hat{S}[j] = s_j^*)$ with s_j^* denoting the true secret

4 Experiment

4.1 Adversary Model

This study investigates a matrix multiplication paradigm of Scloud$^+$ that employs a CPA framework. Within this operational context, the targeted system stores an invisible secret key to execute $C_1' \cdot S$ protocols. The attacker systematically transmits specially engineered ciphertexts to the compromised device, aiming to reconstruct the secret key outputs through leakage analysis. The adversary's operational privileges are formalized as follows:

- **Physical Access:** The attacker possesses unrestricted physical proximity to the victim's computational hardware, enabling non-invasive monitoring and acquisition of power leakage during cryptographic operations.
- **Algorithmic Transparency:** Complete knowledge of the cryptographic scheme's implementation details on the target platform is assumed, including Scloud$^+$ workflow and parameter configurations.

- **Adaptive Query Capacity:** Unrestricted authorization exists to generate arbitrary ciphertext inputs and repeatedly initiate $C_1' \cdot S$ routines on the device under test (DUT).

4.2 Experimental Setup

4.2.1 Equipment Setup

As depicted in Figure 1, the experimental setup for side-channel data acquisition comprises three primary components. The measurement system is built around the ChipWhisperer-Lite platform, the CW308 UFO interface board, and the CW308T-STM32F3 cryptographic target. The ChipWhisperer-Lite module integrates a high-precision measurement engine equipped with a 10-bit analog-to-digital converter (ADC) operating at a sampling rate of 105 MS/s, specifically optimized for capturing power traces during cryptographic operations. This module employs a 24,000-sample circular buffer to facilitate continuous waveform acquisition while concurrently handling host-target communication.

The CW308T-STM32F3 target board, mounted on the CW308 UFO baseboard, serves as the device under test (DUT). It is powered by a 32-bit ARM Cortex-M4 microcontroller, offering a consistent and representative evaluation platform for side-channel analysis. The DUT operates at a nominal clock frequency of 7.37 MHz and employs a synchronous sampling strategy to align measurement precision with cryptographic execution. The cryptographic firmware is compiled using the `arm-none-eabi-gcc` toolchain with maximum compiler optimization enabled (flag -O3), conforming to standard embedded systems development practices.

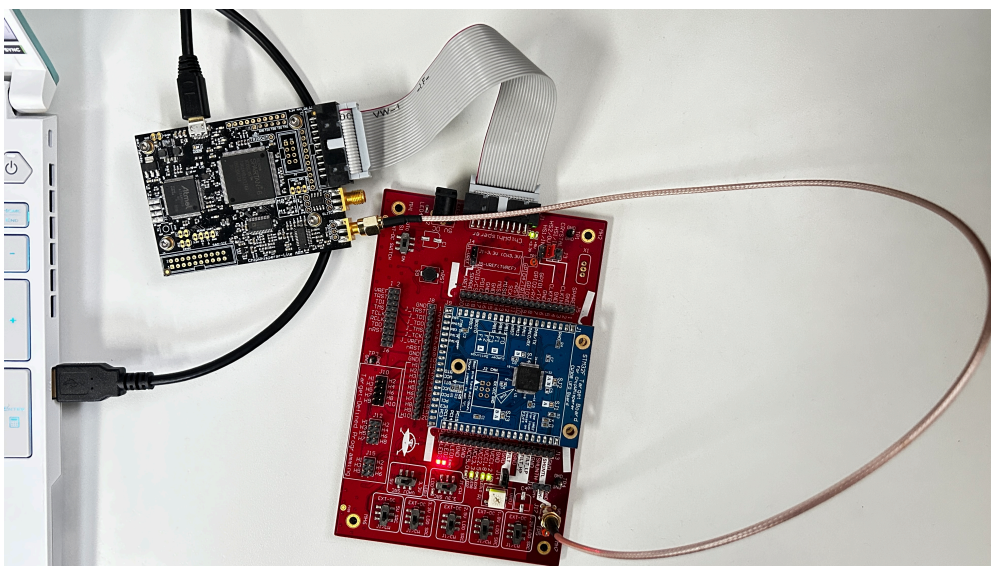


Figure 1 Equipment setup used to capture power traces

In this paper, the `scope.adc.samples` is set to 7000, and the clock frequency is set to 2.5×10^7 Hz. Subsequently, the CS matrix-multiplication algorithm is constructed and uploaded to the target board for CPA.

4.2.2 Data Preparation

In this section, we describe the process of obtaining 1000 sets of C_1 and a fixed key matrix S after encapsulating the Scloud$^+$ source code into a dynamic library. This encapsulation enables us to invoke the function `scloudplus_mul_cs()` to obtain C_1, S, and D reliably.

To ensure experimental reproducibility, we implement the following measures:

1. **Fixed Entropy Source and Personalization String:** By initializing both the entropy source and the personalization string to zero, we guarantee that the output of the random number generator remains deterministic.

2. **Deterministic Key Generation:** The key generation process employs a fixed entropy source and a deterministic random number generator, so that the secret key generated via `func_scloud_kemkeygen()` remains unchanged.

3. **Consistent Unpacking Process:** The unpacking function (`func_scloudplus_unpacksk()`) extracts the secret key matrix S in a consistent manner from the fixed private key, ensuring that S remains the same across different executions.

Thus, regardless of the number of runs, as long as the entropy source and personalization string are kept constant, the generated secret key S remains identical, which is critical for the reproducibility of subsequent experimental results.

The relevant process illustrating this data preparation process as shown in Algorithm 6:

Algorithm 6 Data Preparation for Fixed-S and Multiple C_1 Samples

Output: $S \in \mathbb{Z}_q^{n\times\bar{n}}$, $\{C_1^i\}_{i=1}^{1000} \subset \mathbb{Z}_q^{\bar{m}\times n}$, $\{D^i\}_{i=1}^{1000} \subset \mathbb{Z}_q^{\bar{m}\times\bar{n}}$

Init: Set entropy $E \leftarrow \mathbf{0}$ and personalization $P \leftarrow \mathbf{0}$;
RNG: `func_randombytes_init`(E, P);
KeyGen: Generate $(\text{pk}, \text{sk}) \leftarrow$ `func_scloud_kemkeygen`();
$S \leftarrow$ `func_scloudplus_unpacksk`(sk);
for $i = 1$ **to** 1000 **do**
  $ctx \leftarrow$ `func_scloud_kemencaps`(pk);
  $C_1^i \leftarrow$ `func_scloudplus_unpackc1`(ctx);
  $C_1^i \leftarrow$ `func_scloudplus_decompressc1`(C_1^i);
  $D^i \leftarrow C_1^i \cdot S$;
end
Save: Store $\{C_1^i\}$, S, $\{D^i\}$ to file.

4.2.3 Communication Setup

The C implementation corresponds to the Scloud+ available on the project's GitHub repository. To enable practical evaluation of the matrix multiplication $C_1' \cdot S$ on the CW308T-STM32F3 target board, we design a dedicated communication protocol comprising three operational phases:

1. **Segmented Data Transmission:** The $C_1' \in \mathbb{Z}_q^{\bar{m}\times n}$ and $S \in \mathbb{Z}_q^{n\times\bar{n}}$ matrices undergo packetization into 96 data blocks (100 bytes/block) with cyclic redundancy check (CRC-8) validation. Each block carries 50 matrix elements encoded as little-endian 16-bit integers, ensuring atomic transmission through the `simpleserial` interface.

2. **Matrix Multiplication and Trace Acquisition:** After transmitting the ciphertext (and, if necessary, the key), the target board executes the matrix multiplication $C_1' \cdot S$. Concurrently, the scope is armed to capture the power consumption trace during the multiplication operation. In this study, the scope type is 'OPENADC'.

3. **Result Retrieval:** Once the multiplication completes, the target board returns the processed result along with the captured trace to the PC via the `capture_trace_mul_cs` function as shown in Algorithm 7. This function orchestrates the overall process, including error handling and ensuring data integrity.

Algorithm 7 Power Trace Capture for $C_1 \cdot S$ Multiplication

Input: $Scope$, Target device T, Ciphertext $C_1' \in \mathbb{Z}_q^{\bar{m} \times n}$, Secret key $S \in \mathbb{Z}_q^{n \times \bar{n}}$
Output: Power trace w, result matrix $D = C_1' \cdot S$

```
Segment C_1' || S into 96 blocks of fixed length;
foreach block i in C_1 do
    Encode with CRC and transmit via T.simpleserial_write('C', data);
end
foreach block i in S do
    Encode with CRC and transmit via T.simpleserial_write('S', data);
end
Scope.arm();
Trigger multiplication on target: T.simpleserial_write('D');
Capture trace: w ← Scope.capture();
Read result: D ← T.simpleserial_read('r');
return (w, D);
```

4.3 Leakage Localization Results

In this paper, the aim is to recover 32 secret elements as an example. This necessitates the precise localization of 32 distinct leakage intervals corresponding to the operations $C_1'[i] \cdot S[i]$. Algorithm 3 is executed over a total of 32 iterations, resulting in the acquisition of 32 SoST result files. These files form the basis of the subsequent leakage localization analysis. For example, the interval associated with $C_1'[0] \cdot S[0]$ is delineated by identifying the critical positions in the SoST curves of $C_1'[0]$ and $C_1'[1]$; the leakage region is defined as the range between these two points.

Figure 2 illustrates a representative example of leakage localization. The complete set of 32 SoST curves is arranged into four groups—indices 0–7 (Group 0–7), 8–15 (Group 8–15), 16–23 (Group 16–23), and 24–31 (Group 24–31)—displayed sequentially from top to bottom across four subplots. In each subplot, the x-axis denotes the actual sample indices, while the colored segments indicate regions with significant deviations in SoST values. The intervals between the peaks correspond to the critical leakage boundaries associated with the multiplication operation $C_1'[i] \cdot S[i]$. Such a visualization enables accurate identification of leakage points, which is fundamental to the subsequent side-channel analysis.

By analyzing Figure 2 and applying the dynamic sampling window Γ_j approach described in Section 3.3.1, we can systematically determine reasonable approximate intervals corresponding to each store operation of the form $C_1'[i] \cdot S[i]$. Table 1 summarizes the dynamic leakage intervals for the first 32 operations ($i = 0$ to 31). These sampling windows provide a clear basis for further side-channel analysis. The remaining windows can be extrapolated analogously.

In real-world attacks, a conservatively extended window is often recommended to accommodate clock jitter and measurement noise, thereby enhancing the reliability of key inference. This hardware-guided refinement of Γ_j not only boosts the alignment between hypothetical and real power consumption patterns but also contributes directly to the overall recovery accuracy by mitigating temporal misalignment.

4.4 Iterative Key Recovery: Take 32 key elements as an example

To evaluate the effectiveness of Algorithm 5, we consider the recovery of a subset of the full elements $\mathbf{s}$, which consists of 4800 elements in total. Specifically, we focus on recovering the first 32 elements ($S[0]$ to $S[31]$) as a representative. Each element is associated with a dynamic sampling window Γ_j, configured according to Table 1.

The iterative recovery algorithm is applied to each of the 32 target intervals, and the estimated elements $\hat{s}_j$ are compared with the ground truth values used during trace generation. As a result, all 32 elements are correctly recovered, demonstrating the accuracy of the proposed method.

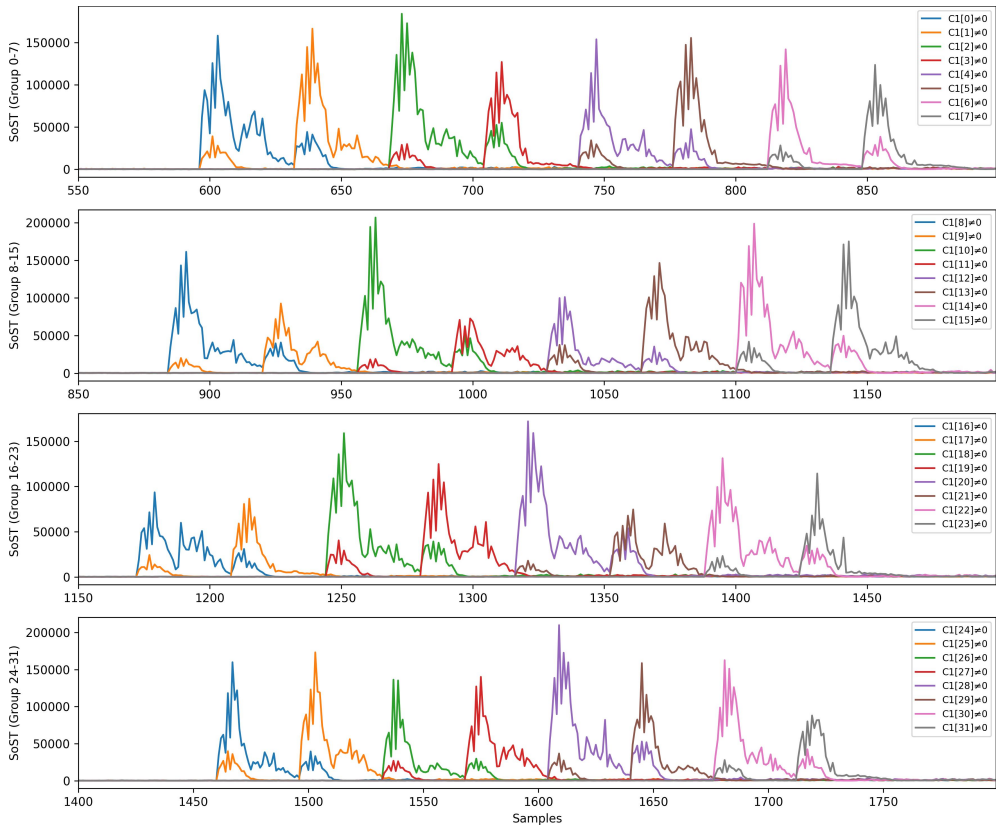


Figure 2 Leakage localization for $\boldsymbol{C_1' \cdot S}$ (indices 0–31)

Table 1 Dynamic sampling window $\boldsymbol{\Gamma_j}$ for each $\boldsymbol{C_1'[i] \cdot S[i]}$ operation

i	**Sampling Interval**	i	**Sampling Interval**
0	600–640	16	1170–1190
1	630–670	17	1210–1250
2	660–710	18	1240–1290
3	700–740	19	1280–1330
4	740–780	20	1320–1360
5	780–820	21	1350–1390
6	820–860	22	1400–1440
7	850–890	23	1430–1460
8	910–940	24	1460–1500
9	920–960	25	1500–1540
10	950–990	26	1540–1570
11	1010–1040	27	1570–1610
12	1030–1070	28	1610–1650
13	1070–1110	29	1640–1680
14	1100–1140	30	1680–1720
15	1120–1170	31	1720–1760

Moreover, we record the minimal number of traces $N_{\min}^{(j)}$ mentioned in Section 3.3.3 to achieve a stable and correct recovery for each element. Figure 3 presents the $N_{\min}^{(j)}$ result, showing that most elements can be reliably recovered with fewer than 15 traces, and some with as few as 2 or 3. This localized success indicates the method's high efficiency and resilience.

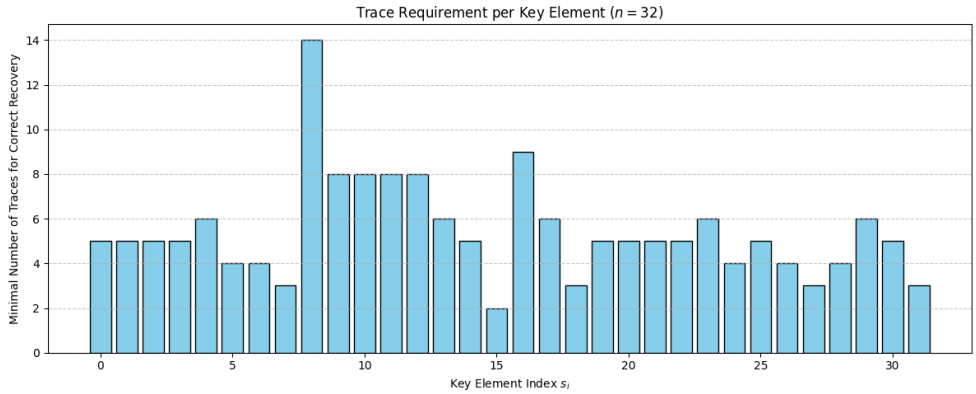


Figure 3 Minimal Traces Required for Recovery of the First 32 Key Elements

It is imperative to emphasise that the emphasis on solely the initial 32 elements is attributable to numerous pivotal considerations. First, the dynamic sampling windows Γ_j are determined through careful observation and manual adjustment based on the leakage profile extracted from the power traces. These windows cannot be simply set as a fixed interval determined empirically and then extrapolated uniformly for all subsequent elements. Instead, the window configuration relies on the inherent characteristics of each computation, requiring iterative tuning and validation. Second, the full key recovery process is inherently a repetitive and iterative procedure. Specifically, Algorithm 5 can process at most 600 elements in a single pass. This limitation stems from the fact that each intermediate value D_i is constructed by the iterative summation of 600 terms of the form $C_i \cdot S_i$. Once the number of elements exceeds 600, the iterative accumulation must be restarted for the next segment. Therefore, recovering the full 4800-element key involves partitioning the process into multiple rounds, each recovering a segment of at most 600 elements. The recovery of the first 32 elements represents one such iteration and serves as a proof-of-concept for the feasibility and performance of the algorithm on each segment.

While the initial focus is on recovering 32 elements, the demonstrated algorithm and performance strongly support the scalability of the approach. By iteratively applying the same recovery procedure, considering the manual adjustment of the dynamic sampling windows and the inherent iterative limit of 600 elements per recovery round, the proposed method can be seamlessly extended to recover all 4800 key elements in practical scenarios.

5 Countermeasure

To mitigate the CPA attacks described in the previous section, we introduce a first-order arithmetic masking scheme, as shown in Algorithm 8. The core idea is to split the original ternary secret matrix S into two additive shares $share_0$ and $share_1$, each defined over $\mathbb{Z}_q$, so that

$$S \;=\; (share_0 + share_1) \mod q\,. \tag{13}$$

During KeyGen, we first generate the mask share $share_0[i] \in \{0, \ldots, 4095\}$. We then obtain the second share as

$$share_1 \;=\; (S - share_0) \mod q, \tag{14}$$

which likewise yields values in $\{0, \ldots, 4095\}$.

By design, $\big(share_0[i] + share_1[i]\big) \bmod q \in \{-1, 0, 1\}$ for every index i, ensuring correctness of the original scheme. However, from an attacker's perspective, this masking significantly increases the complexity of key recovery: instead of guessing a single ternary value per secret element, they must now determine two values $share_0[i]$ and $share_1[i]$ from a domain of size $q = 4096$.

Since the Hamming-weight power model no longer applies directly to the intermediate $C_1' \cdot S$, the leakage observed in registers is statistically independent of any single unmasked secret share. An attacker must recover both shares simultaneously—an infeasible task under the current recursive CPA framework. Consequently, the proposed first-order masking significantly raises the attack complexity and restores resilience against side-channel key recovery.

Similarly, as shown in Algorithm 8, all operations involving the secret matrix S' in the encryption phase are performed using masked shares S_0' and S_1' (where $S' = (S_0' + S_1') \bmod q$). In the decryption phase, the same protection is achieved by refreshing the mask shares $share_0'$ and $share_1'$ for each decryption operation, where $share_0'$ is newly generated and $share_1'$ is derived to maintain the relation $S = (share_0' + share_1') \bmod q$. These modifications preserve functionality while effectively addressing the CPA leakage issues across the entire scheme.

Algorithm 8 Masked Scloud$^+$.PKE Algorithm (First-Order Arithmetic Masking)

$(pk, sk) = \text{Scloud}^+\text{.PKE.KeyGen.Masked}(\alpha)$
  $(\text{seed}_A, r_1, r_2) = F(\alpha) \in \{0,1\}^{128} \times \{0,1\}^{256} \times \{0,1\}^{256}$
  $A = \text{gen}(\text{seed}_A) \in \mathbb{Z}_q^{m \times n}$
  $(share_0, share_1) = \text{MaskedSamplesi}(r_1) \in \mathbb{Z}_q^{n \times \bar{n}} \times \mathbb{Z}_q^{n \times \bar{n}}$ *s.t.* $(share_0 + share_1) \bmod q = S$
  $E = \text{CenBinom}(r_2, (m, \bar{n}), \eta_1)$
  $B = A \cdot share_0 + A \cdot share_1 + E \in \mathbb{Z}_q^{m \times \bar{n}}$
  return $pk = (B, \text{seed}_A)$, $sk = S$ where $S = (share_0 + share_1) \bmod q$
$(C) = \text{Scloud}^+\text{.PKE.Enc.Masked}(pk, m, r)$
  $A = \text{gen}(\text{seed}_A)$
  $(r_1', r_2') = F(r) \in \{0,1\}^{256 \times 2}$
  $(S_0', S_1') = \text{MaskedSamplephi}(r_1') \in \mathbb{Z}_q^{\bar{m} \times m} \times \mathbb{Z}_q^{\bar{m} \times m}$ *s.t.* $(S_0' + S_1') \bmod q = S'$
  $E' = (E_1, E_2) = \text{CenBinom}(r_2', (\bar{m}, n + \bar{n}), \eta_2)$, where $E_1 \in \mathbb{Z}^{\bar{m} \times n}$, $E_2 \in \mathbb{Z}^{\bar{m} \times \bar{n}}$
  $M = \text{MsgEnc}(m) \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$
  $C_1 = S_0' \cdot A + S_1' \cdot A + E_1$, $C_2 = S_0' \cdot B + S_1' \cdot B + E_2 + M$
  $\bar{C}_1 = \left\lfloor \frac{q_1}{q} \cdot C_1 \right\rceil$, $\bar{C}_2 = \left\lfloor \frac{q_2}{q} \cdot C_2 \right\rceil_{\text{odd}}$
  return $C = (\bar{C}_1, \bar{C}_2)$
$(m) = \text{Scloud}^+\text{.PKE.Dec.Masked}((S, C, \beta))$
  $r_3 = F(\beta) \in \{0,1\}^{256}$
  $(share_0', share_1') = \text{MaskedSamplesi}(r_3) \in \mathbb{Z}_q^{n \times \bar{n}} \times \mathbb{Z}_q^{n \times \bar{n}}$ *s.t.* $(share_0' + share_1') \bmod q = S$
  $C_1' = \frac{q}{q_1} \cdot \bar{C}_1$, $C_2' = \frac{q}{q_2} \cdot \bar{C}_2$
  $D = C_2' - C_1' \cdot share_0' - C_1' \cdot share_1' \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$
  return $m = \text{MsgDec}(D) \in \{0,1\}^{l_m}$

6 Conclusion

In this work, we presented a novel CPA framework that targets the ciphertext-key multiplication operations in Scloud$^+$. By combining dynamic sampling windows, instruction-level timing analysis, and iterative correlation maximization, our approach reliably recovers secret coefficients from power

traces. On the Scloud$^+$-128 configuration—using only 32 elements as a case study—we demonstrated high recovery accuracy with fewer than 15 traces per element, thereby exposing a significant leakage channel linked to intermediate Hamming Weight computations. The findings underscore a critical implementation vulnerability, revealing that the leakage associated with intermediate Hamming Weight calculations can be effectively exploited. This not only brings to light potential security concerns for Scloud$^+$ in real-world applications but also points to the necessity of embedding robust side-channel countermeasures during the design and standardization phases of post-quantum cryptosystems.

To address this vulnerability, we further proposed a first-order arithmetic masking countermeasure, which splits the ternary secret S into two subkeys S_0 and S_1 in $\mathbb{Z}_q$. This masking increases the attacker' s search space from $\{-1, 0, 1\}$ to two full-range additive shares per element, effectively thwarting CPA-based leakage.

Our next objective is to adapt and evaluate new attacks against this first-order masked implementation. Specifically, we will investigate tailored strategies to unmask S_0 and S_1 under the new leakage profiles and quantify the residual side-channel risk.

References

[1] SHOR P W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer[J].SIAM Review, 1999, 41(2): 303–332. [DOI: 10.1137/S0036144598347011]

[2] WANG A, ZHENG Z, ZHAO C, QIU Z, ZENG G, YUAN Y, WANG X. Scloud$^+$: An Efficient LWE-based KEM Without Ring/Module Structure[J]. IACR Cryptology ePrint Archive, 2024, Paper 1306. [URL: https://eprint.iacr.org/2024/1306]

[3] AZOUAOUI M, KUZOVKOVA Y, SCHNEIDER T, VAN VREDENDAAL C. Post-Quantum Authenticated Encryption against Chosen-Ciphertext Side-Channel Attacks[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022, 2022(4): 372–396. [DOI: 10.46586/tches.v2022.i4.372-396]

[4] BRIER E, CLAVIER C, OLIVIER F, et al. Correlation power analysis with a leakage model[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2004. Springer Berlin Heidelberg, 2004: 16–29. [DOI: 10.1007/978-3-540-28632-5_2]

[5] KOCHER P C, JAFFE J M, JUN B, et al. Differential power analysis[C]. In: Advances in Cryptology—CRYPTO ' 99. Springer Berlin Heidelberg, 1999: 388–397. [DOI: 10.1007/3-540-48405-1_25]

[6] ATCI A C, BATINA L, GIERLICHS B, et al. Power analysis on NTRU implementations for RFIDs: First results[C]//Proceedings of the 4th Workshop on RFID Security. Budapest, Hungary, 2008: 1–11. [DOI: 10.1007/978-3-540-85230-3_1]

[7] ESPITAU T, FOUQUE P A, GÉRARD B, et al. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongSwan and electromagnetic emanations in microcontrollers[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. Dallas, USA, 2017: 1857–1874. [DOI: 10.1145/3133956.3134028]

[8] AYSU A, TOBAH Y, TIWARI M, et al. Horizontal side-channel vulnerabilities of post-quantum key exchange protocols[C]//Proceedings of the 2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). Washington, USA, 2018: 81–88. [DOI: 10.1109/HST.2018.8430181]

[9] HUANG W L, CHEN J P, YANG B Y. Power analysis on NTRU prime[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2020(1): 123–151. [DOI: 10.46586/tches.v2020.i1.123-151]

[10] HUTTER M, WENGER E. Fast multi-precision multiplication for public-key cryptography on embedded microprocessors[C]//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Berlin, Germany, 2011: 459–474. [DOI: 10.1007/978-3-642-23951-9_31]

[11] FOURNARIS A P, DIMOPOULOS C, KOUFOPAVLOU O. Profiling dilithium digital signature traces for correlation differential side-channel attacks[C]//Proceedings of the International Conference on Embedded Computer Systems. Samos, Greece, 2020: 281–294. [DOI: 10.1109/ICEmS.2020.00052]

[12] Kuo Y T, Takayasu A. A lattice attack on CRYSTALS-Kyber with correlation power analysis[C]. In: Seo H, Kim S, editors. Information Security and Cryptology –ICISC 2023. Lecture Notes in Computer Science, vol. 14561. Singapore: Springer; 2024. p. 202–220. [DOI: 10.1007/978-981-97-1235-9_11]

[13] GOODWILL G, JUN B, JAFFE J, et al. A testing methodology for side-channel resistance validation[C/OL]. In: Proceedings of NIST Non-Invasive Attack Testing Workshop, 2011, 7: 115–136. https://csrc.nist.rip/csrc/media/events/non-invasive-attack-testing-workshop/documents/08_goodwill.pdf

[14] GIERLICHS B, LEMKE-RUST K, PAAR C. Templates vs. stochastic methods[C]. In: Cryptographic Hardware and Embedded Systems (CHES), 2006: 15–29.

[15] LO O, BUCHANAN W J, CARSON D. Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA)[J]. Journal of Cyber Security Technology, 2017, 1(2): 88–107. [DOI: 10.1080/23742917.2016.1231523]

[16] LI Yan-Bin, ZHU Jia-Jie, TANG Ming, ZHANG Huan-Guo. Power analysis attacks for lattice-based cryptography[J]. Journal of Cryptologic Research, 2023, 10(6): 609–627. [DOI: 10.11897/SP.J.1016.2023.00331]

[17] TIAN J, WEI Y, XU D, et al. Fast Scloud$^+$: A Fast Hardware Implementation for the Unstructured LWE-based KEM - Scloud$^+$[J/OL]. Cryptology ePrint Archive, 2025, Paper 2025/497. Available: https://eprint.iacr.org/2025/497.

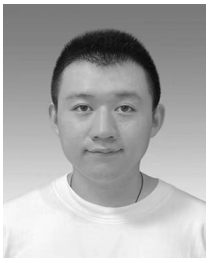
Author Information



白航宇 (2002–)，河南南阳人，硕士生在读．主要研究领域为密码算法的安全性分析，硬件安全．
bhy@mail.ustc.edu.cn



黄帆 (1994–)，安徽合肥人，博士生在读．主要研究领域为密码算法的安全性分析，硬件安全．
lanplush@mail.ustc.edu.cn



段晓林 (1991–)，安徽合肥人，博士生在读．主要研究领域为密码算法的安全性分析，硬件安全．
duanxl@mail.ustc.edu.cn



胡红钢 (1978–)，四川彭州人，教授，博士生导师．主要研究领域为密码，编码与网络安全．
hghu2005@ustc.edu.cn

针对 Scloud$^+$ 的高效密钥恢复：基于相关能量分析攻击

白航宇[1]，黄　帆[1]，段晓林[1]，胡红钢[1,2]

1. 中国科学技术大学 网络空间安全学院, 合肥 230026
2. 合肥国家实验室, 合肥 230088

通信作者: 胡红钢, E-mail: hghu2005@ustc.edu.cn

摘　要: Scloud$^+$ 作为新一代后量子密钥封装机制，通过非结构化带错误学习（LWE）问题与三元密钥编码技术的结合，在消除传统环代数结构约束的同时实现了高效的格密码运算。尽管该算法在理论安全性层面已进行了严格的形式化证明，其实际部署仍面临安全威胁。近年来，侧信道攻击尤其是相关能量分析（CPA）已成为评估密码算法抗攻击能力的重要手段。针对上述背景，本文首次针对其核心运算模块——密文-密钥矩阵乘法的物理实现安全性展开系统性研究，提出一种融合指令级时序分析的相关能量分析框架。首先，为更有效地定位泄漏点，本文采用基于多分组 Welch's t 检验思想的 SoST（Sum of Squared T-values）模型分析密文加载操作的汉明权重泄漏特征，结合处理器微架构级指令流水线行为建模，利用动态采样窗口定位每个密文—密钥乘法过程中的关键泄漏区间。随后，针对三元密钥的特点，我们采用迭代恢复策略：在预定义的扫描顺序下，通过三元候选值集合 $\{-1, 0, 1\}$ 和部分中间和的迭代计

算，使用汉明重量（HW）模型构造假设泄露向量；再结合对每个动态采样窗口内各采样点与能量迹数据的 Pearson 相关分析，利用轨迹数稳定性检测（trace-count stabilization），确定每个密钥元素的最优估计值，从而实现全密钥向量的逐步重构。以全体 4800 个密钥元素为恢复目标的实验中，我们以前 32 个密钥元素为示例，对其恢复效果进行详细分析。实验结果显示，前 32 个密钥元素能以 100% 的准确率恢复，且每个元素所需的最少能量迹数均不超过 15 条，这验证了所提方法在局部区域内的高效性和稳定性，且该结果可通过迭代计算扩展至全体密钥元素。为抵御此类 CPA 攻击，本文进一步设计并实现了一阶算术掩码防护方案，通过将原始三元密钥拆分为两个子密钥，将攻击者的密钥猜测范围从 $\{-1,0,1\}$ 扩展至 $\{0,\ldots,4095\}^2$，显著提升了侧信道抗性。研究结果表明，Scloud$^+$ 在物理实现层面存在侧信道防御漏洞，亟需在标准化进程中纳入相关防护措施。

关键词： 侧信道攻击；相关能量分析；后量子密码；Scloud$^+$

中图分类号： TP309.7 **文献标识码：** A **DOI**: 10.13868/j.cnki.jcr.000XXX

中文引用格式: 白航宇, 黄帆, 段晓林, 胡红钢. 针对 Scloud$^+$ 的高效密钥恢复：基于相关能量分析攻击[J]. 密码学报 (中英文), 2025, X(X): 1–21. [DOI: 10.13868/j.cnki.jcr.000XXX]