Fine-Grained Residual Interval Screening Enhanced Least Square for the Secret Key Recovery of Dilithium

Kailong Shi[1,3], Hailong Zhang[2], Donddai Lin[1,3], Zichen Wang[4]

[1] Key Laboratory of Cyberspace Security Defense,
Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Control and Computer Engineering,
North China Electric Power University
[3] School of Cyber Security, University of Chinese Academy of Sciences
[4] Department of Electrical and Computer Engineering, School of Engineering and Applied Science, George Washington University
zhanghailong@ncepu.edu.cn

Abstract. In practice, the amount of side channel leakage related to the random polynomial generation of Dilithium can be limited. In this case, the coefficients of the random polynomial may not be recovered accurately, which then makes the secret key recovery with least square a difficult problem. Therefore, how to recover the secret key used by Dilithium with noisy equations becomes a meaningful concern. In light of this, the fine-grained residual interval screening is proposed to enhance the ability of least square to recover the secret key used by Dilithium. The core idea is to estimate the distributions of the residuals related to both correct and erroneous equations computed with the least square recovered candidate secret key in a profiling scenario. Then, according to the distribution difference of the residuals related to two types of equations, an interval can be screened. Note that a majority of erroneous equations are out of the screened interval while a majority of correct equations are in the screened interval. Therefore, least square can be used to recover a more accurate candidate secret key polynomial with equations in the screened interval. By iterating the process several times, the secret key polynomial can be successfully recovered. The efficiency of the proposed technique is verified with power traces measured from the Dilithium reference implementation running on a Cortex-M4 processor. The evaluation results show that only several hundreds of power traces are enough to recover the secret key used by Dilithium.

Keywords: Post-Quantum Cryptography · Dilithium · Side-Channel Resistance · Secret Key Recovery · Residual Interval Screening

1 Introduction

In the scenario of quantum computing, the mathematical problems (e.g., the large integer factorization [20] and the discrete logarithm solving [11]) that the

security of traditional public-key algorithms based on may not be hard to solve anymore. The reason is that the algorithm proposed by Shor [21] can solve these problems in polynomial time on a quantum computer. Therefore, traditionally deployed public-key algorithms may not be secure anymore, and it is necessary to design public-key algorithms that are secure in the quantum computing scenario. In light of this, the National Institute of Standards and Technology (NIST) has initiated the standardization process for Post-Quantum Cryptography (PQC) in 2017. In 2022, NIST announced three PQC digital signature algorithms, i.e., Dilithium [9], Falcon [22], and SPHINCS+ [3]. In fact, according to the PQC standardization status report released in July 2022 [1], Dilithium can be a good choice in terms of efficiency, implementation cost, and security. Therefore, NIST decided to prioritize the standardization of Dilithium under the name Module Lattice-Based Digital Signature Algorithm (ML-DSA)[16]. Note that Dilithium follows the Lyubashevsky–Peikert–Regev (LPR) design paradigm [14]. Besides, its security relies on the Module Learning With Errors (MLWE) problem [15].

In practice, Dilithium can be implemented into a chip to achieve the goal of information security. When a chip is in operation, different types of side-channel leakage can be measured and exploited to recover the secret key used by Dilithium, posing a serious threat to the practical security of Dilithium. In light of this, it is necessary to investigate the side-channel resistance of Dilithium in practice. In fact, since Dilithium was proposed at TCHES 2018 [9], researchers in the cryptographic engineering community have fully realized the importance of analyzing its side-channel resistance. Consequently, different types of side-channel attacks have been proposed to recover the secret key used by Dilithium, and representatives of them are summarized as follows.

- First, NTT-based polynomial multiplication can be chosen as the target. Here, NTT is the abbreviation of number theory transform, which is fast Fourier transform in the finite field. In detail, Chen et al. at ICCD 2021 [7] as well as Tosun and Savas at IEEE TIFS 2024 [25] exploited the leakage of $\hat{\mathbf{c}}\hat{\mathbf{s}}_1$ to recover the private key $\mathbf{s}_1$ of Dilithium reference implementation running on a Cortex-M4 processor using correlation power analysis (CPA). Recently, Fan et al. at IEEE TIFS 2025 [12] exploited both the leakage of $\hat{\mathbf{c}}\hat{\mathbf{s}}_1$ and $\hat{\mathbf{A}}\hat{\mathbf{y}}_1$ to recover $\mathbf{s}_1$ of Dilithium reference implementation on Cortex-M4 with the so called Multivariate Template Attack (MTA).
- Second, secret key unpacking can be chosen as the target. In detal, Wang et al. [28] used deep learning assisted power analysis to recover the secret keys $\mathbf{s}_1$ and $\mathbf{s}_2$ of a fast Dilithium implementation on Cortex-M4 by exploiting the side-channel leakage related to the unpacking of $\mathbf{s}_1$ and $\mathbf{s}_2$. Recently, Wang et al. [27] showed that with the side-channel leakage related to the unpacking of $\mathbf{t}_0$, deep-learning based profiling power analysis can be used to partially recover the least significant bit of $\mathbf{t}_0$. Then, the number of signatures required for the reconstruction of $\mathbf{t}_0$ is reduced by roughly half, and the new $\mathbf{t}_0$ reconstruction method enhances the efficiency of recovering $\mathbf{s}_1$.
- Third, the computation of the low-order bits of $\mathbf{w}_0$ can be chosen as the target. In detail, at TCHES 2023, Berzati et al. [4] used the side-channel

leakage related to the computation of $\mathbf{w}_0$ to recover $\mathbf{w}_0$ with template attack (TA). Then, the value of $\mathbf{w}_0$ can be used to build linear equations related to $\mathbf{s}_1$, and the solving of which can recover $\mathbf{s}_1$ used by the Dilithium reference implementation running on a Cortex-M4 processor.
- Fourth, the random polynomial generation can be chosen as the target. In detail, Marzougui et al. at SAC 2022 [26], Qiao et al. at IEEE TIFS 2023 [18], and Fan et al. at IEEE TIFS 2024 [13] used the side-channel leakage related to the random polynomial generation to recover coefficients of the random polynomial with profiling attacks. Then, coefficients of the random polynomial can be used to build linear equations related to $\mathbf{s}_1$, and the solving of which with least square or linear programming can recover $\mathbf{s}_1$.
- Fifth, NTT can be chosen as the target of side-channel attacks. In detail, at CHES 2017, Primas et al. [17] used the side-channel leakage related to NTT to recover the secret key of an ring learning with errors encryption scheme on an ARM Cortex-M4F processor with the soft analytic side-channel attacks. Then, at PQCRYPTO 2023, Steffen et al. [24] used the side-channel leakage related to NTT to recover $\mathbf{s}_1$ and $\mathbf{s}_2$ used by an FPGA implementation of Dilithium.

It can be seen that existing works chose to recover $\mathbf{s}_1$ used by Dilithium. The reason is that according to [19] and [6], the knowledge of $\mathbf{s}_1$ is adequate to forge a valid signature for an arbitrary message with a high probability. Besides, it can be noticed that the generation of the random polynomial is a main target of side-channel attacks. However, when the generation of the random polynomial is chosen as the target, the amount of side-channel leakage available is limited. The reason is that the polynomial is randomly generated. In this case, one can only use the side-channel leakage contained in one power trace to recover coefficients of the random polynomial. Because the amount of side-channel leakage available in one power trace is limited, coefficients of the random polynomial may not be correctly recovered. In this case, linear equations related to $\mathbf{s}_1$ may be noisy, and $\mathbf{s}_1$ may not be correctly recovered with least square or linear programming. Therefore, how to recover $\mathbf{s}_1$ with noisy equations becomes a meaningful concern. In light of this, the fine-grained residual interval screening technique is proposed to enhance the ability of least square to recover $\mathbf{s}_1$. In detail, the contributions of this work are summarized as follows.

- We empirically estimate the distributions of the residuals related to both correct and erroneous equations computed with the least square recovered candidate secret key polynomial in a profiling scenario. Note that whether or not coefficients of the random polynomial are correctly recovered can be known in a profiling scenario. If the coefficients related to an equation are correctly recovered, the equation is referred to as a correct equation; else, it is referred to as an erroneous equation. Then, according to the distribution difference of the residuals related to two types of equations, an interval can be screened. In detail, a majority of erroneous equations are out of the screened interval while a majority of correct equations are in the screened interval. In

this case, least square can be used to recover a more accurate candidate with equations in the screened interval. By iterating the process several times, the secret key polynomial can be successfully recovered.
- The effectiveness of the fine-grained residual interval screening technique to enhance the efficiency of least square to recover $\mathbf{s}_1$ of Dilithium is verified in the real scenario. In detail, power traces related to the generation of the random polynomial of Dilithium reference implementation on Cortex M4 are used. The evaluation results show that MTA proposed by Fan et al. [13] can recover coefficients of the random polynomial with the accuracy of 7.08% for Dilithium 2 and the accuracy of 4.15% for Dilithium 3. In this case, screening least square technique (SLST) proposed by Fan et al. [13] cannot recover $\mathbf{s}_1$ used by Dilithium, while fine-grained residual interval screening enhanced least square can recover $\mathbf{s}_1$ used by Dilithium 2 with 270 power traces and $\mathbf{s}_1$ used by Dilithium 3 with 400 power traces. Therefore, the evaluation results show that fine-grained residual interval screening enhanced least square can be used to more accurately evaluate the practical security of Dilithium.

The rest of this paper is organized as follows. The notation, the signature algorithm of Dilithium, and the least square used to recover $\mathbf{s}_1$ of Dilithium are introduced in Section 2. Section 3 provides a detailed description of the proposed fine-grained residual interval screening technique. In Section 4, the effectiveness of the fine-grained residual interval screening technique to enhance the efficiency of least square to recover $\mathbf{s}_1$ of Dilithium is verified in the real scenario. Finally, Section 5 concludes the paper.

2 Preliminary

First, notations used in this work are introduced. Second, details about the signature algorithm of Dilithium are introduced. Third, the least square used to recover $\mathbf{s}_1$ of Dilithium is introduced.

2.1 Notations

Let $\mathbb{Z}_q$ denote the ring of integers modulo q. Then, $\mathbb{Z}_q[x]$ denotes the set of polynomials with coefficients in $\mathbb{Z}_q$. We can define $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ as the polynomial ring modulo $x^n + 1$. In Dilithium q is set to $2^{23} - 2^{13} + 1$ and n is set to 256. $\mathbf{A}$ denotes a matrix with k rows and l columns, and each element in $\mathbf{A}$ is a polynomial in R_q. The value of (k, l) varies with the version of Dilithium. In detail, for Dilithium 2, $(k, l) = (4, 4)$; for Dilithium 3, $(k, l) = (6, 5)$; and for Dilithium 5, $(k, l) = (8, 7)$. $\mathbf{y}$ is a random polynomial vector with l elements, and each element in $\mathbf{y}$ is a polynomial in R_q. The coefficients of an element in $\mathbf{y}$ are in the range of $[-\gamma_1, \gamma_1]$, and γ_1 can be set to 2^{17}, 2^{19}, and 2^{19} in Dilithium 2, 3, and 5, respectively. $\mathbf{c} \in R_q$ is the challenge polynomial, which has exactly τ coefficients equal to ± 1 and the rest coefficients of $\mathbf{c}$ are equal to 0. In Dilithium 2, 3, and 5, τ can be set to 39, 49, and 60, respectively. The secret key $\mathbf{s}_1$ is a

vector with l elements while the secret key $\mathbf{s}_2$ is a vector with k elements, and each element in $\mathbf{s}_1$ and $\mathbf{s}_2$ is a polynomial in R_q. The coefficients of elements in $\mathbf{s}_1$ and $\mathbf{s}_2$ are in the range of $[-\eta, \eta]$. In Dilithium 2, 3, and 5, η can be set to 2, 4, and 2, respectively. $\hat{\mathbf{w}}$, $\hat{\mathbf{A}}$, $\hat{\mathbf{y}}$, $\hat{\mathbf{c}}$, and $\hat{\mathbf{s}}_1$ denote $\mathbf{w}$, $\mathbf{A}$, $\mathbf{y}$, $\mathbf{c}$, and $\mathbf{s}_1$ in the NTT domain, respectively. Besides, $\hat{\mathbf{w}}$, $\hat{\mathbf{A}}$, $\hat{\mathbf{y}}$, $\hat{\mathbf{c}}$, and $\hat{\mathbf{s}}_1$ can be obtained from $\mathbf{w}$, $\mathbf{A}$, $\mathbf{y}$, $\mathbf{c}$, and $\mathbf{s}_1$ through the NTT function, respectively. $\|\mathbf{z}\|_\infty$ is the ∞-norm of $\mathbf{z}$, whose value equals to the largest absolute value of coefficients of $\mathbf{z}$.

2.2 The Signature Algorithm of Dilithium

Dilithium is consisted of three algorithms, i.e. the key generation algorithm, the signing algorithm and the verification algorithm. In this work, the side-channel leakage related to the random polynomial generation is used to recover $\mathbf{s}_1$ used by Dilithium. Here, the random polynomial generation is a step in the signing algorithm of Dilithium. Therefore, only the signing algorithm of Dilithium is introduced, which is shown in Algorithm 1.

Algorithm 1 Sign(sk, M)

Require: $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0), M$
Ensure: $sig = (\mathbf{z}, \mathbf{h}, \tilde{\mathbf{c}})$
$\hat{\mathbf{A}} \in R_q^{k \times l} \leftarrow \text{ExpandA}(\rho)$
$\mu \in \{0,1\}^{512} \leftarrow H(tr \parallel M)$
$k \leftarrow 0, (\mathbf{z}, \mathbf{h}) \leftarrow \perp$
$\rho' \in \{0,1\}^{512} \leftarrow H(K \parallel \mu)$ ▷ or $\rho' \leftarrow \{0,1\}^{512}$ for randomized signing
while $(\mathbf{z}, \mathbf{h}) = \perp$ **do**
  $\mathbf{y} \in S_{\gamma_1}^l \leftarrow \text{ExpandMask}(\rho', k)$
  $\hat{\mathbf{y}} \leftarrow \text{NTT}(\mathbf{y}), \hat{\mathbf{w}} \leftarrow \hat{\mathbf{A}}\hat{\mathbf{y}}$
  $\mathbf{w} \leftarrow \text{NTT}^{-1}(\hat{\mathbf{w}})$
  $\mathbf{w}_1 \leftarrow \text{HighBits}_q(\mathbf{w}, 2\gamma_2)$
  $\tilde{\mathbf{c}} \in \{0,1\}^{256} \leftarrow H(\mu \parallel \mathbf{w}_1)$
  $\mathbf{c} \leftarrow \text{SampleInBall}(\tilde{\mathbf{c}})$
  $\hat{\mathbf{c}} \leftarrow \text{NTT}(\mathbf{c}), \hat{\mathbf{s}}_1 \leftarrow \text{NTT}(\mathbf{s}_1)$
  $\hat{\mathbf{u}} \leftarrow \hat{\mathbf{c}}\hat{\mathbf{s}}_1$
  $\mathbf{z} \leftarrow \mathbf{y} + \text{NTT}^{-1}(\hat{\mathbf{u}})$
  $\mathbf{r_0} \leftarrow \text{LowBits}_q(\mathbf{w} - \mathbf{cs}_2, 2\gamma_2)$
  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **or** $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then**
    $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$
  else
    $\mathbf{h} \leftarrow \text{MakeHint}_q(-\mathbf{ct}_0, \mathbf{w} - \mathbf{cs}_2 + \mathbf{ct}_0, 2\gamma_2)$
    if $\|\mathbf{ct}_0\|_\infty \geq \gamma_2$ **or** the # of 1's in $\mathbf{h}$ is greater than ω **then**
      $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$
    end if
  end if
  $k \leftarrow k + l$
end while
return $sig = (\mathbf{z}, \mathbf{h}, \tilde{\mathbf{c}})$

According to Algorithm 1, in order to sign a message M using the secret key sk, one should compute $\hat{\mathbf{w}} = \hat{\mathbf{A}}\hat{\mathbf{y}}$. Then, $\mathbf{w}$ can be obtained by applying the inverse NTT function. The challenge polynomial $\mathbf{c}$ can be derived from the high-order bits $\mathbf{w}_1$ of $\mathbf{w}$. Then, a part of the signature, i.e. $\mathbf{z}$, can be computed as $\mathbf{z} = \mathbf{y} + \mathbf{c}\mathbf{s}_1$. Finally, in order to ensure that the signature does not leak information about the private key and for correctness purposes, several boundary checks are performed. If one check fails, a new signature candidate should be generated; otherwise, $\mathbf{z}$ and $\tilde{\mathbf{c}}$, together with the hint information $\mathbf{h}$, are output as the signature.

2.3 Least Square

For a noisy linear system $\mathbf{C}\mathbf{s}_1 \approx \mathbf{b}$ (hereafter abbreviated as $(\mathbf{C}, \mathbf{b})$), the solution of $\mathbf{s}_1$ can be obtained using least square. Here, least square aims to find a solution $\mathbf{s}_1^*$ that minimizes the ℓ_2 (Euclidean) norm of the residual $\mathbf{e} = \mathbf{C}\mathbf{s}_1 - \mathbf{b}$ [5]. In fact, according to line 14 of Algorithm 1, one has $\mathbf{z} = \mathbf{y} + \mathbf{c}\mathbf{s}_1$, where $\mathbf{z}$ and $\mathbf{c}$ are known while $\mathbf{y}$ and $\mathbf{s}_1$ are unknown. With the side-channel leakage contained in one power trace related to the generation of $\mathbf{y}$, coefficients of $\mathbf{y}$ may not be correctly recovered. If $\mathbf{y}^*$ is used to denote the random polynomial recovered with side-channel attacks, we have $\mathbf{z} \approx \mathbf{y}^* + \mathbf{c}\mathbf{s}_1$, which can be seen as a noisy linear system. In detail, in order to compute $\mathbf{c}\mathbf{s}_1$, $\mathbf{c}$ can be expressed as a matrix $\mathbf{C}$ composed of coefficients of $\mathbf{c}$. In detail, we have

$$\mathbf{C} = \begin{bmatrix} c_0 & -c_{255} & -c_{254} & \cdots & -c_1 \\ c_1 & c_0 & -c_{255} & \cdots & -c_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{255} & c_{254} & c_{253} & \cdots & c_0 \end{bmatrix},$$

where c_i are i^{th} coefficient of $\mathbf{c}$. Besides, we have $\mathbf{b} = \mathbf{z} - \mathbf{y}^*$. Then, least square can be used to recover $\mathbf{s}_1$. Note that the probability of least square to recover $\mathbf{s}_1$ depends on the accuracy of $\mathbf{y}^*$ and the number of equations contained in the noisy linear system. On one hand, the lower the accuracy of $\mathbf{y}^*$, the lower the probability of least square to recover $\mathbf{s}_1$; on the other hand, the larger the number of equations contained in the noisy linear system, the higher the probability of least square to recover $\mathbf{s}_1$.

3 Fine-Grained Residual Interval Screening Enhanced Least Square

At IEEE TIFS 2024, Fan et al. [13] proposed the SLST technique, which can be used to recover $\mathbf{s}_1$ with noisy linear system $\mathbf{C}\mathbf{s}_1 \approx \mathbf{b}$. In this technique, a candidate of $\mathbf{s}_1$ should be recovered with ($\mathbf{C}$,$\mathbf{b}$), which can be denoted as $\mathbf{s}_1^*$. Then, the absolute value of the residual $\mathbf{e}$, i.e., $|\mathbf{e}| = |\mathbf{C}\mathbf{s}_1^* - \mathbf{b}|$ is computed. If the average of elements in $|\mathbf{e}|$ is denoted as $m_{|\mathbf{e}|}$, equations with absolute values of residuals in the interval $[0, m_{|\mathbf{e}|}]$ are screened. In this case, $(\mathbf{C}, \mathbf{b})$ is

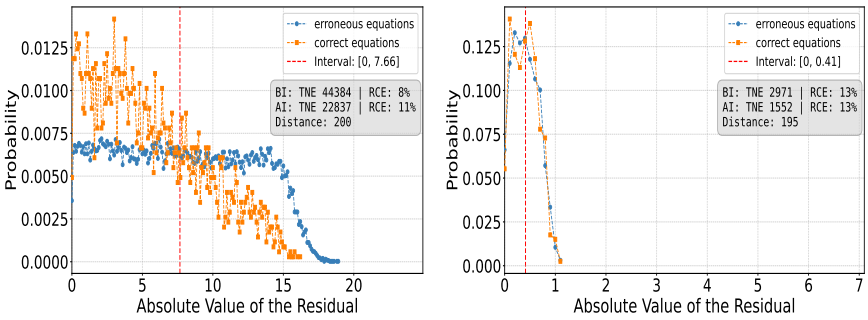
updated for the next iteration. After repeating this iteration several times, $\mathbf{s}_1$ can be recovered from $\mathbf{s}_1^*$ by using the *Round*() function. Here, the funciton of *Round*() is to round a value to its nearest integer. The idea of SLST is to choose equations with small residuals to make it easier to recover $\mathbf{s}_1$. However, according to the evaluation result, it can be infeasible to recover $\mathbf{s}_1$ used by the Dilithium reference implementation running on a Cortex-M4 processor with SLST.

In order to find out the reason for the failure of SLST, the distributions of the residuals of correct and erroneous equations are estimated in the simulated scenario. Note that in the simulated scenario values of different parameters are under full control. In this case, the distributions of the residuals of two types of equations can be accurately estimated. In detail, power traces related to the random polynomial generation of the Dilithium 2 reference implementation on Cortex-M4 are simulated with Python. Here, the simulated power traces contain side-channel leakages related to the processing of $\mathbf{y}$, the unsigned number vector $\mathbf{x}$, and the random byte vector $\mathbf{a}$, where $\mathbf{x}$ and $\mathbf{a}$ are two vectors closely related to $\mathbf{y}$. For more details about $\mathbf{x}$ and $\mathbf{a}$, please refer to [13]. For each coefficient of $\mathbf{y}$, $\mathbf{x}$ and $\mathbf{a}$, the number of interesting points is set to 10 empirically to make MTA perform well. Here, interesting points are samples in power traces related to the processing of the target intermediate value. The leakage at each interesting point is composed of two parts, i.e., the signal part which depends on the processed value and the noise part which can be irrelevant to the processed value. Because the leakage of a value on Cortex-M4 follows the Hamming weight model [13], the signal part is simulated under the Hamming weight model. Comparatively, the noise part should be simulated under the multivariate normal distribution assumption. Besides, the correlation coefficient of the noise part at different interesting points is empirically set to 0.2 to simulate the cross correlation of the noise part at different interesting points. Finally, the signal-to-noise ratio (SNR) of the simulated leakage is set to 5, 0.35, and 0.15 to approximate the SNR of the leakage related to the processing of $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{a}$ on Cortex-M4.

Subsequently, MTA proposed by Fan et al. [13] is used to recover $\mathbf{y}$. Here, under the assumption that high bits of $\mathbf{y}$ are equal to high bits of $\mathbf{z}$ with a high probability, only low bits of $\mathbf{y}$ need to be recovered with MTA. In detail, low 5, 6, and 7 bits of $\mathbf{y}$ can be recovered with MTA. Then, together with high bits of $\mathbf{z}$, $\mathbf{y}$ can be recovered. In detail, in order to recover low bits of $\mathbf{y}$, 50000 power traces are used in the profiling phase to characterize the leakage properties related to the processing of $\mathbf{y}$, $\mathbf{x}$ and $\mathbf{a}$. Then, one trace can be used in the attack phase of MTA to recover low bits of $\mathbf{y}$. Together with high bits of $\mathbf{z}$, $\mathbf{y}$ can be recovered. By repeating the attack phase 2000 times with 2000 randomly generated power traces, the accuracy to recover $\mathbf{y}$ can be estimated. In detail, the accuracy to recover $\mathbf{y}$ in three cases can be 7.84%, 4.63% and 2.33%, respectively.

With the value of $\mathbf{y}^*$, the linear system $(\mathbf{C}, \mathbf{b})$ can be built. Note that one value of $\mathbf{y}^*$ is related to a linear system of 256 linear equations. If multiple values of $\mathbf{y}^*$ are available, the number of linear equations in the linear system can be multiples of 256. In our case, in order to estimate the distributions of the residuals of correct and erroneous equations, 180 power traces are used, and MTA is used

to recover low 5 bits of $\mathbf{y}$. Then, with 180 values of $\mathbf{z}$, 180 values of $\mathbf{y}^*$ can be obtained. In theory, the linear system is composed of 180*256=46080 linear equations. With the linear system built, a candidate $\mathbf{s}_1^*$ of $\mathbf{s}_1$ can be recovered with least square. Then, one can compute the absolute value of the residual $\mathbf{e}$, i.e. $|\mathbf{e}| = |\mathbf{C}\mathbf{s}_1^* - \mathbf{b}|$. Note that in a profiling scenario, 180 values of $\mathbf{y}$ are known. In this case, 46080 equations in the linear system can be divided into two groups, one containing equations whose values in $\mathbf{b}$ are correct, while the other containing equations whose values in $\mathbf{b}$ are erroneous. Finally, the distributions of the residuals of correct and erroneous equations can be estimated.



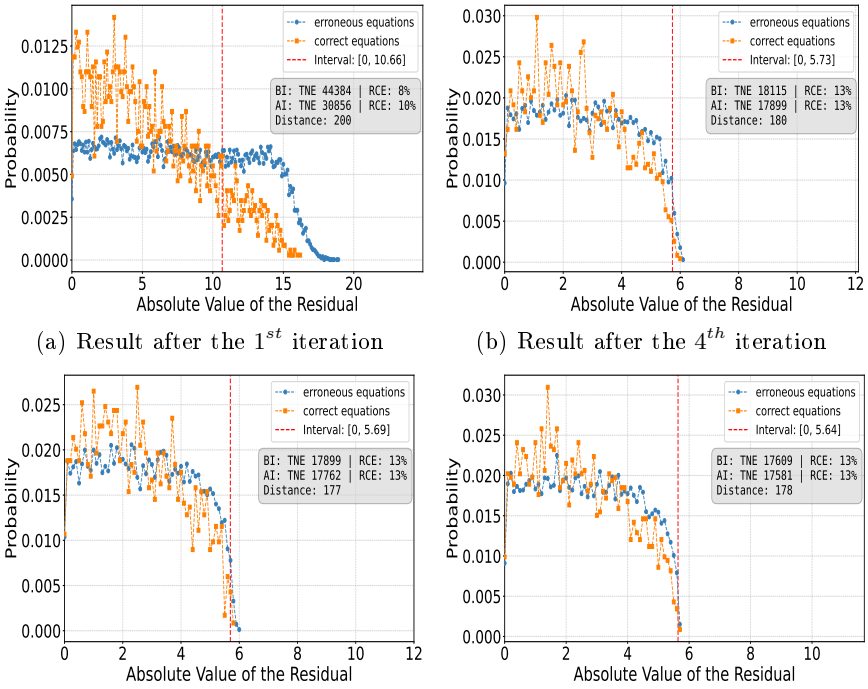
(a) Result after the 1st iteration (b) Result after the 5th iteration

Fig. 1. Performances of SLST after the 1st iteration and the 5th iteration.

In Fig. 1, BI is the abbreviation of before iteration, AI is the abbreviation of after iteration, TNE is the abbreviation of total number of equations, RCE is the abbreviation of rate of correct equations, Distance denote the number of coefficients of $Round(\mathbf{s}_1^*)$ that are not equal to coefficients of $\mathbf{s}_1$. Fig. 1(a) shows the distributions of the residuals of correct and erroneous equations after the 1st iteration, while Fig. 1(b) shows the distributions of the residuals of correct and erroneous equations after the 5th iteration. By observing Fig. 1(a), the difference between the distributions of the residuals of correct and erroneous equations can be significant. In this case, by choosing equations with residuals in the interval $[0,m_{\mathbf{e}}]$, RCE increases from 8% to 11%. However, according to Fig 1(b), the difference between the distributions of the residuals of correct and erroneous equations can be minor. In this case, by choosing equations with residuals in the interval $[0,m_{\mathbf{e}}]$, RCE does not increase anymore. Therefore, the Distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ can still be large, and it is infeasible to recover $\mathbf{s}_1$ with SLST. Finally, note that for some Hamming weights that occur with a small probability, the related templates are not built. Then, coefficients of $\mathbf{y}$ related to these Hamming weights are not recovered. Therefore, the total number of equations shown in Fig. 1 is 44384 rather than 46080.

Overall, the reason for the failure of SLST is that the number of correct equations in the chosen interval is small and the rate of correct equations is

low. In light of this, we decide to enlarge the interval to increase the number of correct equations while maintaining the rate of correct equations. According to Fig. 1(a), we empirically choose equations whose residuals are in the interval of $[0, m_{\mathbf{e}} + 3]$. Then, the performances of least square after the 1st iteration, the 4th iteration, the 8th iteration and the 12th iteration can be evaluated.



(a) Result after the 1st iteration

(b) Result after the 4th iteration

(c) Result after the 8th iteration

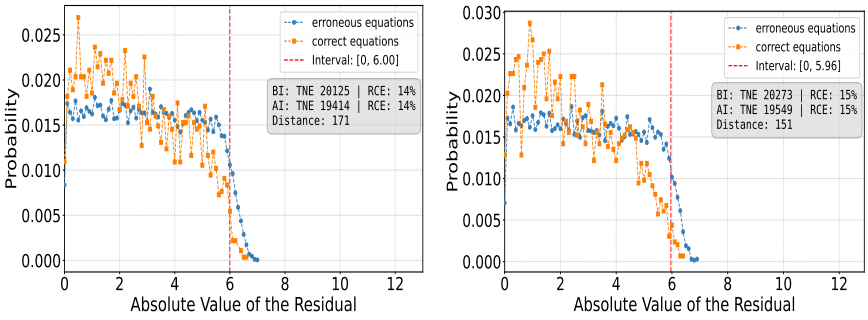
(d) Result after the 12th iteration

Fig. 2. Performances of least square after different times of iteration with equations whose residuals are in the interval of $[0, m_{\mathbf{e}} + 3]$.

According to Fig.2(a), the number of correct equations is increased and the rate of correct equations is maintained. Therefore, after the 4th iteration, the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ can be decreased to 180. However, further repeating the iteration does not help to decrease the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$. According to Fig. 2(b) to Fig. 2(d), the distributions of the residuals of correct and erroneous equations are similar, and RCE does not increase anymore. In light of this, the iteration should be stopped after the 4th iteration.

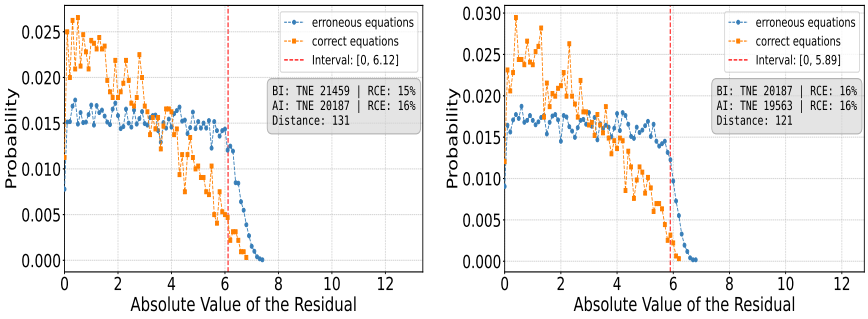
Then, considering that after the 4th iteration the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ is reduced to 180, we come up with the idea of using $\mathbf{s}_1^*$ as the new candidate of $\mathbf{s}_1$ to repeat the above iteration. Here, we refer to the above four times iteration as a Type-I global iteration. We believe that with this more

accurate candidate of $\mathbf{s}_1$, the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ can be further reduced by repeating the global iteration. The evaluation results after different times of Type-I global iteration are shown as follows.



(a) Result after the 2nd global iteration

(b) Result after the 3rd global iteration



(c) Result after the 4th global iteration

(d) Result after the 5th global iteration

Fig. 3. Performances of least square after different times of Type-I global iteration with equations whose residuals are in the interval of $[0, m_{\mathbf{e}} + 3]$.

According to Fig. 3, by repeating Type-I global iteration three times, the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ is reduced further. However, after the 4th Type-I global iteration, the decrease of the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ is not obvious. Therefore, it seems that for equations with residuals in the interval of $[0, m_{\mathbf{e}} + 3]$, the number of Type-I global iteration should be set to 4.

It can be seen from Fig. 3 that after several times of Type-I global iteration, the residuals of correct equations are distributed in a more compact interval. In light of this, we shrink the screening interval from $[0, m_{\mathbf{e}} + 3]$ to $[0, m_{\mathbf{e}} + 2]$ and repeat Type-I global iteration several times with more accurate candidates of $\mathbf{s}_1$, and the evaluation results can be seen in Fig. 9 shown in Appendix A. It can be seen that after four times of Type-I global iteration, the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ is reduced to 103, and further repeating Type-I global iteration does not help to reduce the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$. Therefore, the

number of Type-I global iteration should be set to 4. Still, because the residuals of correct equations after four times of Type-I global iteration are distributed in a more compact interval, we shrink the screening interval from $[0, m_{\mathbf{e}} + 2]$ to $[0, m_{\mathbf{e}} + 1]$ and repeat Type-I global iteration several times with more accurate candidates of $\mathbf{s}_1$. Fig. 10 in Appendix A shows that by performing Type-I global iteration four times with more accurate candidates of $\mathbf{s}_1$, the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ is reduced to 90, and further repeating Type-I global iteration does not help to reduce the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$. Therefore, the number of Type-I global iteration is set to 4. Following the similar observation, we shrink the screening interval from $[0, m_{\mathbf{e}} + 1]$ to $[0, m_{\mathbf{e}}]$ and repeat Type-I global iteration several times with more accurate candidates of $\mathbf{s}_1$. Fig. 11 shows that after four times of Type-I global iteration, the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ is only reduced from 90 to 85, which can be not obvious. Therefore, it is not advisable to follow this residual interval screening routine further.

In light of this, we use the more accurate candidate of $\mathbf{s}_1$ recovered after four times of Type-I global iteration with equations whose residuals are in the interval of $[0, m_{\mathbf{e}}]$ to estimate the distributions of the residuals of correct and erroneous equations, and the evaluation result is shown in Fig. 4.

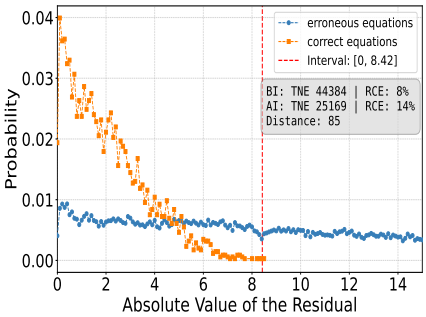
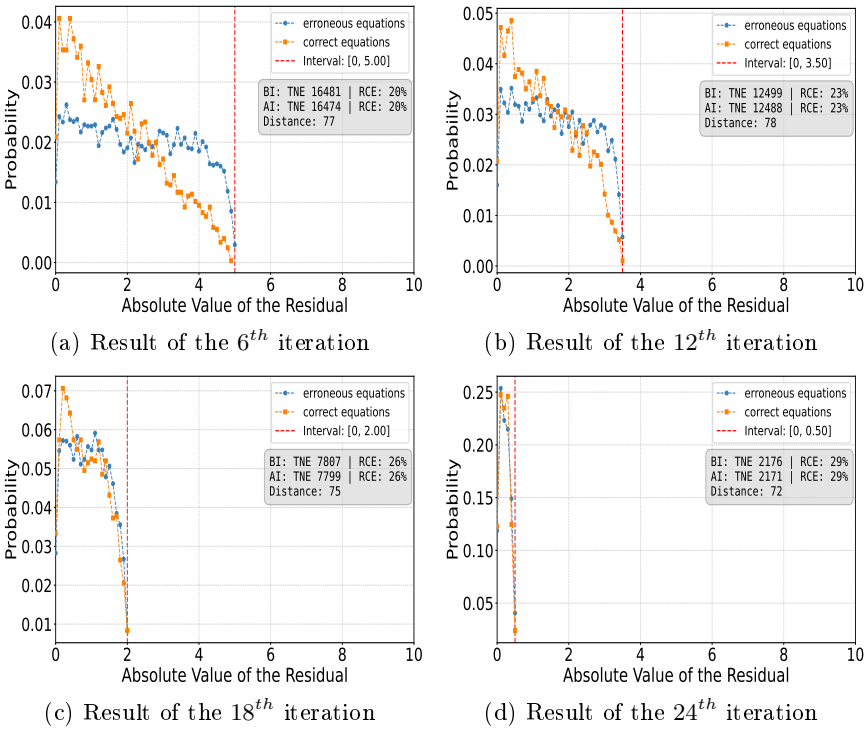


Fig. 4. Distributions of the residuals of correct equations and erroneous equations with the more accurate candidate of $\mathbf{s}_1$.

It can be seen that in this case the residuals of correct equations are mainly in the interval of $[0, 5]$. In light of this, the screening interval should be set to $[0, 5]$. Then, using equations with residuals in the interval of $[0, 5]$, least square can be repeatedly used 6 times to recover a more accurate candidate of $\mathbf{s}_1$. The reason for repeating least square 6 times with equations whose residuals are in the interval of $[0, 5]$ is that further repeating least square in this case does not help to reduce the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$. As can be seen from Fig. 5(a), the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ in this case is reduced to 77. Then, because the residuals of most correct equations are in the interval of $[0, 3.5]$, least square can be repeatedly used 6 times to recover a more accurate candidate of $\mathbf{s}_1$ with equations whose residuals are in this interval. Still, the reason for repeating

least square 6 times is that further repeating least square in this case does not help to reduce the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$. According to Fig. 5(b), although the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ is not reduced in this case, the value of RCE is increased. Similarly, least square can be repeatedly used 6 times with equations whose residuals are in the intervals of $[0, 2]$ and $[0, 0.5]$ in order to further reduce the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$. According to Fig. 5(c) and Fig. 5(d), the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ after 18 and 24 times of iteration is reduced from 75 to 72 and RCE is increased from 26% to 29%.



(a) Result of the 6th iteration

(b) Result of the 12th iteration

(c) Result of the 18th iteration

(d) Result of the 24th iteration

Fig. 5. Performances of least square after different times of iteration with equations whose residuals are in the intervals of $[0, 5]$, $[0, 3.5]$, $[0, 2]$ and $[0, 0.5]$.

If the above 24 times of iteration are referred to as Type-II global iteration, the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ is expected to be reduced by iterating Type-II global iteration several times with more accurate candidates of $\mathbf{s}_1$.

Table 1. Distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ after 15 times of Type-II global iteration.

Time	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Distance	72	61	50	40	28	16	14	10	8	8	7	7	6	6	6

Indeed, Table 1 shows that it is very effective to reduce the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ by repeating Type-II global iteration several times. However, $\mathbf{s}_1$ can still be not recovered even if Type-II global iteration is repeated further. In light of this, the distributions of the residuals of correct and erroneous equations are estimated with the candidate of $\mathbf{s}_1$ recovered after the 15th global iteration, and the estimation result is shown in Fig. 6.

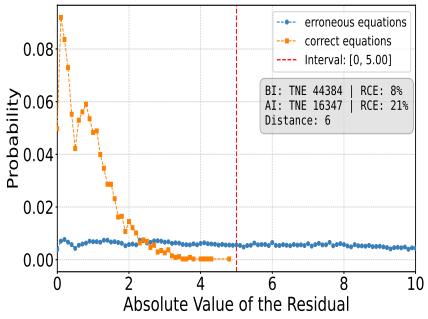


Fig. 6. Distributions of the residuals of correct equations and erroneous equations with the candidate of $\mathbf{s}_1$ recovered after the 15th Type-II global iteration.

It can be seen that after the 15th Type-II global iteration, the residuals of correct equations are mainly in the interval of $[0, 2]$. In light of this, equations with residuals in the interval of $[0, 2]$ should be used to recover a more accurate candidate of $\mathbf{s}_1$. In detail, least square should be repeatedly used 6 times with equations whose residuals are in the interval of $[0, 2]$ to recover a more accurate candidate of $\mathbf{s}_1$. Here, the reason to repeat least square 6 times with equations whose residuals are in the interval of $[0, 2]$ is that it does not help to reduce the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ anymore if further iterations are performed. According to Fig. 12(a) shown in Appendix A, after the 6th iteration, RCE increases to 38%. Then, it seems that the residuals of most correct equations are in the interval of $[0, 1.5]$. In light of this, least square is repeatedly used 6 times with equations whose residuals are in the interval of $[0, 1.5]$. Still, the reason to repeat least square 6 times here is that it does not help to reduce the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ anymore if further iterations are performed. According to Fig. 12(b) shown in Appendix A, after the 12th iteration, RCE increases to 43%. Similarly, least square is repeatedly used 6 times with equations whose residuals are in the intervals of $[0, 1]$ and $[0, 0.5]$. According to Fig. 12(c) and Fig. 12(d) shown in Appendix A, RCE increases to 49% after the 18th iteration, while RCE increases to 54% after the 24th iteration. Besides, after the 24th iteration, the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ is reduced to 2.

Still, if the above 24 times of iteration together are referred to as Type-III global iteration, Fig. 12 in Appendix A shows that after a Type-III global iteration $\mathbf{s}_1$ can still be not recovered. In light of this, Type-III global iteration

can be repeated several times with more accurate candidates of $\mathbf{s}_1$ to recover $\mathbf{s}_1$. In fact, the distance between $Round(\mathbf{s}_1^*)$ and $\mathbf{s}_1$ after different times of Type-III global iteration is evaluated, and the evaluation result is shown in Table 2. It can be seen that $\mathbf{s}_1$ can be recovered after only 2 times of Type-III global iteration.

Table 2. Secret key $\mathbf{s}_1$ recovery result after 20 times of Type-III global iteration.

Time	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Distance	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Overall, the fine-grained residual interval screening enhanced least square is composed of three phases. In phase 1, 4 times of Type-I global iteration should be performed. During each Type-I global iteration, 4 times of iteration whose residuals are in the intervals of $[0, m_{\mathbf{e}} + 3]$, $[0, m_{\mathbf{e}} + 2]$, $[0, m_{\mathbf{e}} + 1]$, and $[0, m_{\mathbf{e}}]$ should be performed in order. Then, 15 times of Type-II global iteration should be performed in phase 2. During each Type-II global iteration, 6 times of iteration with equations whose residuals are in the intervals of $[0, 5]$, $[0, 3.5]$, $[0, 2]$, and $[0, 0.5]$ should be performed in order. Finally, 20 times of Type-III global iteration should be performed in phase 3. During each global iteration, 6 times of iteration with equations whose residuals are in the intervals of $[0, 2]$, $[0, 1.5]$, $[0, 1]$, and $[0, 0.5]$ should be performed in order. Therefore, the overall procedure can be summarized in Algorithm 2.

Note that the intervals shown in Algorithm 2 are screened with the samples of $\mathbf{y}^*$ obtained by performing MTA on the low 5 bits of $\mathbf{y}$ used in Dilithium 2. In fact, under the same style, the intervals screened with the samples of $\mathbf{y}^*$ obtained by performing MTA on the low 6 or 7 bits of $\mathbf{y}$ used in Dilithium 2 can be obtained. The results show that in three cases the screened intervals can be about the same. Besides, the intervals screened with the samples of $\mathbf{y}^*$ obtained by performing MTA on the low 5, 6, and 7 bits of $\mathbf{y}$ used in Dilithium 3 and 5 can also be obtained. In detail, the intervals screened with the samples of $\mathbf{y}^*$ obtained by performing MTA on the low 5, 6, and 7 bits of $\mathbf{y}$ used in Dilithium 3 can be about the same, while the intervals screened with the samples of $\mathbf{y}^*$ obtained by performing MTA on the low 5, 6, and 7 bits of $\mathbf{y}$ used in Dilithium 5 can be about the same. However, when $\mathbf{s}_1$ of Dilithium 3 is targeted, the value of V in phase 2 of Algorithm 2 should be started from 8.5 and ended to 0.5 with the step of -1, while the value of V in phase 3 of Algorithm 2 should be started from 3.5 and ended to 0.5 with the step of -0.5. Besides, when $\mathbf{s}_1$ of Dilithium 5 is targeted, the value of V in phase 2 of Algorithm 2 should be started from 6.5 and ended to 0.5 with the step of -1.5, while the value of V in phase 3 of Algorithm 2 should be started from 3 and ended to 0.5 with the step of -0.5. Then, $\mathbf{s}_1$ used by Dilithium 3 and 5 can be recovered with Algorithm 2. Note that according to the running result, on a PC with AMD Ryzent 9 7845HX CPU (2.5GHz), 16G memory and NVIDIA GeForce RTX 4060 GPU graphics card, less than 3 minutes can be enough to recover $\mathbf{s}_1$ with Algorithm 2.

Algorithm 2 fined-grained residual interval screening enhanced least square

Define LEASTSQUARES$(\mathbf{C}, \mathbf{b})$ as solving $\mathbf{s}_1$ from $\mathbf{C}\mathbf{s}_1 \approx \mathbf{b}$ with least square.
Define SCREEN$(\mathbf{C}', \mathbf{b}', \mathbf{s}_1^*, T)$ as screening equations in $(\mathbf{C}', \mathbf{b}')$ whose residuals are smaller than T and obtaining the updated linear system $(\mathbf{C}', \mathbf{b}')$.

Require: $(\mathbf{C}, \mathbf{b})$
Ensure: $\mathbf{s}_1$

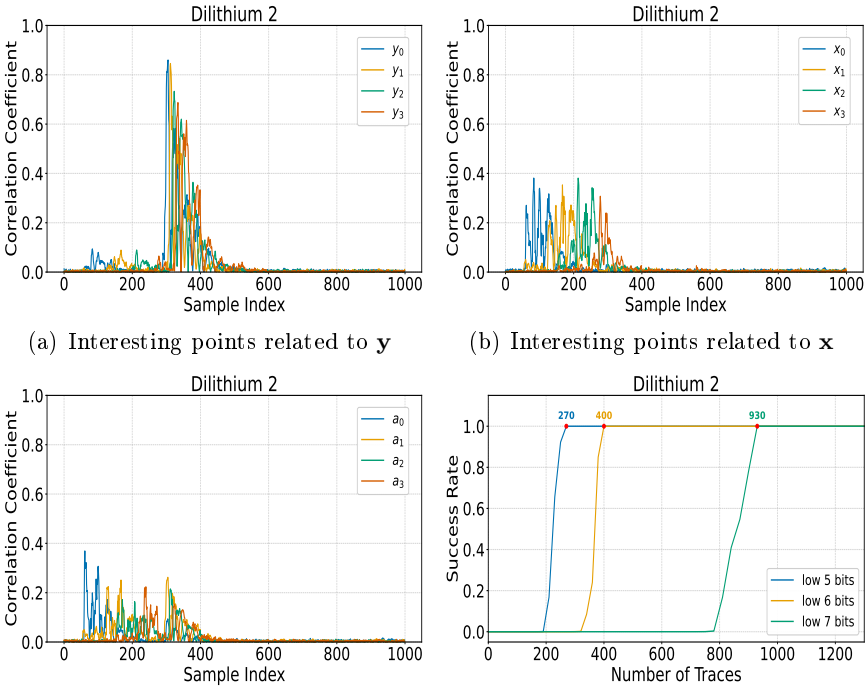
$\mathbf{s}_1^* \leftarrow$ LEASTSQUARES$(\mathbf{C}, \mathbf{b})$ ▷ Obtain the candidate of $\mathbf{s}_1$
▷ **phase 1**
for $V \leftarrow 3$ **down to** 0 **step** -1 **do**
  for $i \leftarrow 0$ **to** 3 **do**
    $\mathbf{C}' = \mathbf{C}, \mathbf{b}' = \mathbf{b}$
    $\mathbf{e} \leftarrow |\mathbf{C}'\mathbf{s}_1^* - \mathbf{b}'|, \quad T \leftarrow m_{\mathbf{e}} + V$ ▷ $T = m_{\mathbf{e}} + V$
    $(\mathbf{C}', \mathbf{b}\prime) \leftarrow$ SCREEN$(\mathbf{C}', \mathbf{b}', \mathbf{s}_1^*, T)$
    for $j \leftarrow 0$ **to** 3 **do**
      $\mathbf{s}_1^* \leftarrow$ LEASTSQUARES$(\mathbf{C}', \mathbf{b}')$
      $\mathbf{e} \leftarrow |\mathbf{C}'\mathbf{s}_1^* - \mathbf{b}'|, \quad T \leftarrow m_{\mathbf{e}} + V$
      $(\mathbf{C}', \mathbf{b}') \leftarrow$ SCREEN$(\mathbf{C}', \mathbf{b}', \mathbf{s}_1^*, T)$ ▷ $T = m_{\mathbf{e}} + V$
    end for
  end for
end for
▷ **phase 2**
for $i \leftarrow 0$ **to** 14 **do**
  $\mathbf{C}' = \mathbf{C}, \mathbf{b}' = \mathbf{b}$
  $(\mathbf{C}', \mathbf{b}') \leftarrow$ SCREEN$(\mathbf{C}', \mathbf{b}', \mathbf{s}_1^*, 5)$ ▷ $T = 5$
  for $V \leftarrow 5$ **down to** 0.5 **step** -1.5 **do**
    for $j \leftarrow 0$ **to** 5 **do**
      $\mathbf{s}_1^* \leftarrow$ LEASTSQUARES$(\mathbf{C}', \mathbf{b}')$
      $(\mathbf{C}', \mathbf{b}') \leftarrow$ SCREEN$(\mathbf{C}', \mathbf{b}', \mathbf{s}_1^*, V)$ ▷ $T = V$
    end for
  end for
end for
▷ **phase 3**
for $i \leftarrow 0$ **to** 19 **do**
  $\mathbf{C}' = \mathbf{C}, \mathbf{b}' = \mathbf{b}$
  $(\mathbf{C}', \mathbf{b}') \leftarrow$ SCREEN$(\mathbf{C}', \mathbf{b}', \mathbf{s}_1^*, 2)$ ▷ $T = 2$
  for $V \leftarrow 2$ **down to** 0.5 **step** -0.5 **do**
    for $j \leftarrow 0$ **to** 5 **do**
      $\mathbf{s}_1^* \leftarrow$ LEASTSQUARES$(\mathbf{C}', \mathbf{b}')$
      $(\mathbf{C}', \mathbf{b}') \leftarrow$ SCREEN$(\mathbf{C}', \mathbf{b}', \mathbf{s}_1^*, V)$ ▷ $T = V$
    end for
  end for
end for

$\mathbf{s}_1 \leftarrow$ ROUND$(\mathbf{s}_1^*)$
return $\mathbf{s}_1$

4 Efficiency Evaluation in the Real Scenario

In the real scenario, power traces are measured during the random polynomial generation of the Dilithium reference implementation on a Cortex-M4 processor. In detail, the target device is ChipWhisperer CW308 STM32F415 running at a clock frequency of 7.38 MHz. The oscilloscope used is a Pico 5244D operating in the 12-bit mode with the time base set to 10, which corresponds to a sampling rate of approximately 8.93 MHz. During the measurement, the low-pass filter is used to decrease the noise level, with the parameter of DC–5MHz 50ohm. The amplifier is not used during the measurement.

First, in order to recover $\mathbf{s}_1$ used by Dilithium 2, MTA [13] should be used to recover the coefficients of $\mathbf{y}$. Here, ρ-test proposed at EUROCRYPT 2016 [10] is used to choose the interesting points related to the processing of a coefficient of $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{a}$ with 50000 power traces. The results are shown from Fig. 7(a) to Fig. 7(c). It can be seen that the leakages related to the processing of $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{a}$ are clearly visible. Then, 20, 20 and 10 interesting points related to the processing of a coefficient of $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{a}$ are chosen, and the leakages at interesting points are used in MTA to recover the coefficients of $\mathbf{y}$.



(a) Interesting points related to $\mathbf{y}$

(b) Interesting points related to $\mathbf{x}$

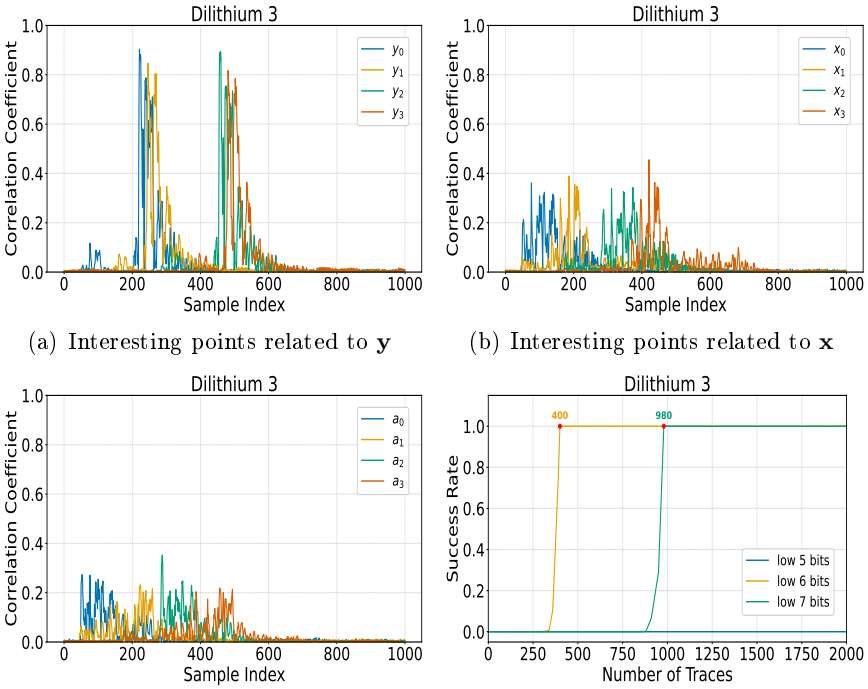
(c) Interesting points related to $\mathbf{a}$

(d) Success rate to recover $\mathbf{s}_1$

Fig. 7. Using the side-channel leakages related to the processing of $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{a}$ to recover $\mathbf{s}_1$ used by Dilithium 2.

During the profiling phase, 50,000 power traces are used to characterize the leakage property related to the generation of $\mathbf{y}$. In the attack phase, using the leakage contained in one power trace, low 5, 6, and 7 bits of a coefficient of $\mathbf{y}$ can be recovered with MTA. Together with the high bits of a coefficient of $\mathbf{z}$, a coefficient of $\mathbf{y}$ can be recovered. The accuracy to recover a coefficient of $\mathbf{y}$ with MTA is evaluated by repeating the attack phase of MTA 2000 times. According to the evaluation results, the accuracy to recover a coefficient of $\mathbf{y}$ can be 7.08%, 4.09%, and 2.14%, respectively. Then, with noisy samples of $\mathbf{y}$, Algorithm 2 can be used to recover $\mathbf{s}_1$ used by Dilithium 2, and the evaluation result is shown in Fig. 7(d). Here, the success rate metric [23] is used to evaluate the efficiency to recover $\mathbf{s}_1$. It can be seen that in the best case only 270 power traces can be enough to recover $\mathbf{s}_1$ used by Dilithium 2.

Second, in order to recover $\mathbf{s}_1$ used by Dilithium 3,the leakages related to the processing of $\mathbf{y}$ are used to recover the coefficients of $\mathbf{y}$. In detail, ρ-test is used to choose the interesting points related to the processing of $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{a}$ with 55000 power traces. The results are shown in Fig. 8(a) to Fig. 8(c). Then, 20, 20 and 10 interesting points related to the processing of a coefficient of $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{a}$ are chosen, and the leakages at interesting points are used in MTA to recover $\mathbf{y}$.



(a) Interesting points related to $\mathbf{y}$

(b) Interesting points related to $\mathbf{x}$

(c) Interesting points related to $\mathbf{a}$

(d) Success rate to recover $\mathbf{s}_1$

Fig. 8. Using the side-channel leakages related to the processing of $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{a}$ to recover $\mathbf{s}_1$ used by Dilithium 3.

In detail, 55000 power traces are used in the profiling phase of MTA to characterize the leakage properties related to the processing of $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{a}$. Then, the leakages contained in one power trace can be used in the attack phase of MTA to recover the coefficients of $\mathbf{y}$. Still, we evaluate the accuracy to recover a coefficient of $\mathbf{y}$ when low 5, 6, and 7 bits of a coefficient of $\mathbf{y}$ are targeted in MTA, and the evaluation results show that the accuracy can be 5.27%, 4.15%, and 2.35% in three cases. Then, with noisy samples of $\mathbf{y}$, Algorithm 2 is used to recover $\mathbf{s}_1$ used by Dilithium 3. According to Fig. 8(d), only 400 power traces can be enough to recover $\mathbf{s}_1$ used by Dilithium 3 in the best case. However, if the samples of $\mathbf{y}^*$ are obtained by recovering the low 5 bits of $\mathbf{y}$ with MTA, $\mathbf{s}_1$ used by Dilithium 3 cannot be recovered. The reason is that the distributions of the residuals of correct and erroneous equations in this case can be about the same. In this case, erroneous equations cannot be screened out to recover $\mathbf{s}_1$ used by Dilithium 3.

Finally, the performance of fine-grained residual interval screening enhanced least square is compared with its counterparts. Note that Marzougui et al. at SAC 2022 [26], Qiao et al. at IEEE TIFS 2023 [18] and Fan et al. at IEEE TIFS 2024 [13] all used the leakage related to the generation of $\mathbf{y}$ to recover $\mathbf{s}_1$ used by Dilithium. First, Marzougui et al. at SAC 2022 [26] used 780000 power traces to recover $\mathbf{s}_1$ used by Dilithium 2. Second, Qiao et al. at IEEE TIFS 2023 [18] used 10000, 35000, and 18000 power traces respectively to recover $\mathbf{s}_1$ used by Dilithium 2, 3, and 5. Third, Fan et al. at IEEE TIFS 2024 [13] claimed that 38, 40, 39 power traces can be enough to recover $\mathbf{s}_1$ used by Dilithium 2, 3, and 5. However, in our attack scenario, $\mathbf{s}_1$ used by Dilithium 2 and 3 cannot be recovered with even 1000 power traces. Comparatively, this work shows that $\mathbf{s}_1$ used by Dilithium 2 and 3 can be recovered with only 270 and 400 power traces. In light of this, fine-grained residual interval screening enhanced least square can be quite effective to recover $\mathbf{s}_1$ used by the Dilithium reference implementation on a Cortex-M4 processor with the leakage related to the generation of $\mathbf{y}$.

5 Conclusions

In this work, fine-grained residual interval screening is proposed to enhance the ability of least square to recover $\mathbf{s}_1$ used by Dilithium. In this technique, by observing the difference between the distributions of residuals of correct and erroneous equations, a residual interval can be chosen. Because most of the correct equations are in the chosen interval while most of the erroneous equations are not in the chosen interval, by using equations with residuals in the chosen interval, a more accurate candidate of $\mathbf{s}_1$ can be recovered. By iterating the above process several times, $\mathbf{s}_1$ can be recovered. The effectiveness of the technique is verified in the real scenario. Note that when a masked Dilithium implementation (e.g., [2] or [8]) is targeted, MTA can be used to recover each share of $\mathbf{y}$. Then, $\mathbf{y}$ can be recovered by combing each share of $\mathbf{y}$. However, in this case, the accuracy to recover $\mathbf{y}$ may be even lower. In light of this, the usability of this technique to recover $\mathbf{s}_1$ used by a masked Dilithium implementation is left as a future work.

Acknowledgements This work is supported in part by the National Natural Science Foundation of China 62272451, the Beijing Natural Science Foundation L234079.

References

1. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Liu, Y.K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D.: Status report on the third round of the nist post-quantum cryptography standardization process. Tech. rep., National Institute of Standards and Technology (2022)
2. Azouaoui, M., Bronchain, O., Cassiers, G., Hoffmann, C., Kuzovkova, Y., Renes, J., Schneider, T., Schönauer, M., Standaert, F.X., van Vredendaal, C.: Protecting dilithium against leakage: Revisited sensitivity analysis and improved implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(4), 58–79 (2023)
3. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The sphincs+ signature framework. In: ACM SIGSAC Conference on Computer and Communications Security – CCS 2019. pp. 2129–2146 (2019)
4. Berzati, A., Calle Viera, A., Chartouny, M., Madec, S., Vergnaud, D., Vigilant, D.: Exploiting intermediate value leakage in dilithium: A template based approach. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(4), 188–210 (2023)
5. Björck, Å.: Least squares methods. Handbook of numerical analysis **1**, 465–652 (1990)
6. Bruinderink, L.G., Pessl, P.: Differential fault attacks on deterministic lattice signatures. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(3), 21–43 (2018)
7. Chen, Z., Karabulut, E., Aysu, A., Ma, Y., Jing, J.: An efficient non-profiled side-channel attack on the crystals-dilithium post-quantum signature. In: International Conference on Computer Design – ICCD 2021. pp. 583–590 (2021)
8. Coron, J.S., Gérard, F., Trannoy, M., Zeitoun, R.: Improved gadgets for the high-order masking of dilithium. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(4), 110–145 (2023)
9. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(1), 238–268 (2018)
10. Durvaux, F., Standaert, F.: From improved leakage detection to the detection of points of interests in leakage traces. In: Advances in Cryptology – EUROCRYPT 2016. Lecture Notes in Computer Science, vol. 9665, pp. 240–262 (2016)
11. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985)
12. Fan, H., Zhang, H., Wang, Y., Wang, W., Zhang, H., Yuan, Q.: Multivariate template attack against ntt-based polynomial multiplication of dilithium. IEEE Transactions on Information Forensics and Security **20**, 8570–8582 (2025)
13. Fan, H., Zhang, H., Wang, Y., Wang, W., Zhu, Y., Zhang, H., Yuan, Q.: Screening least square technique assisted multivariate template attack against the random polynomial generation of dilithium. IEEE Transactions on Information Forensics and Security **19**, 7118–7132 (2024)

14. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology – EUROCRYPT 2010. Lecture Notes in Computer Science, vol. 6110, pp. 1–23 (2010)

15. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 700–718 (2012)

16. National Institute of Standards and Technology (NIST): Module-Lattice-Based Digital Signature Standard (ML-DSA), FIPS 204 (2024)

17. Primas, R., Pessl, P., Mangard, S.: Single-trace side-channel attacks on masked lattice-based encryption. In: Cryptographic Hardware and Embedded Systems – CHES 2017. Lecture Notes in Computer Science, vol. 10529, pp. 513–533 (2017)

18. Qiao, Z., Liu, Y., Zhou, Y., Ming, J., Jin, C., Li, H.: Practical public template attacks on crystals-dilithium with randomness leakages. IEEE Transactions on Information Forensics and Security **18**, 1–14 (2023)

19. Ravi, P.V., Jhanwar, M.P., Howe, J., Chattopadhyay, A., Bhasin, S.: Side-channel assisted existential forgery attack on dilithium - a nist pqc candidate. IACR Cryptology ePrint Archive **2018**, 821:1–21 (2018)

20. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **26**(1), 96–99 (1983)

21. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review **41**(2), 303–332 (1999)

22. Soni, D., Basu, K., Nabeel, M., Aaraj, N., Manzano, M., Karri, R.: Falcon. In: Hardware Architectures for Post -- Quantum Digital Signature Schemes, pp. 31–41 (2021)

23. Standaert, F., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Advances in Cryptology – EUROCRYPT 2009. Lecture Notes in Computer Science, vol. 5479, pp. 443–461 (2009)

24. Steffen, H., Land, G., Kogelheide, L., Güneysu, T.: Breaking and protecting the crystal: Side channel analysis of dilithium in hardware. In: Post Quantum Cryptography — PQCrypto 2023. Lecture Notes in Computer Science, vol. 14154, pp. 688–711 (2023)

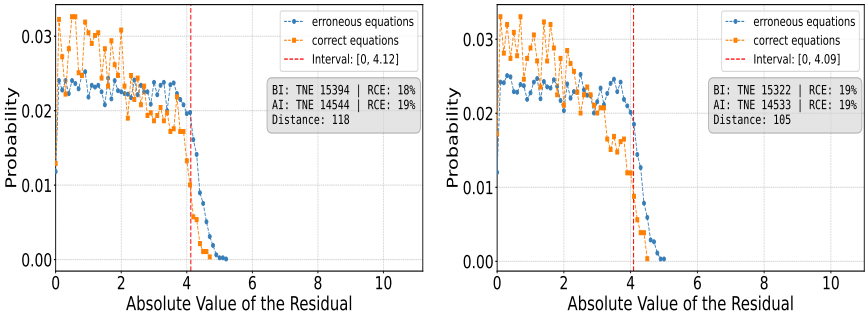
25. Tosun, T., Savas, E.: Zero-value filtering for accelerating non-profiled side-channel attack on incomplete NTT-based implementations of lattice-based cryptography. IEEE Transactions on Information Forensics and Security **19**, 3353–3365 (2024)

26. Ulitzsch, V.Q., Marzougui, S., Tibouchi, M., Seifert, J.P.: Profiling side-channel attacks on dilithium: A small bit-fiddling leak breaks it all. In: Selected Areas in Cryptography – SAC 2022. Lecture Notes in Computer Science, vol. 13742, pp. 3–32 (2022)

27. Wang, R., Gärtner, J., Dubrova, E.: Decompressing dilithium's public key with fewer signatures using side channel analysis. In: International Symposium on Multiple-Valued Logic – ISMVL 2025. pp. 135–140 (2025)

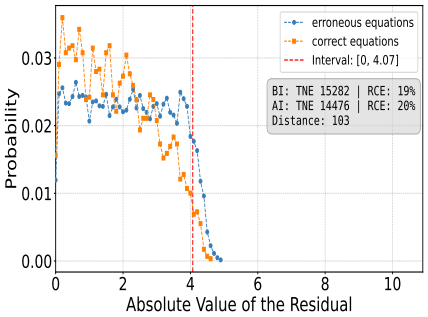
28. Wang, R., Ngo, K., Gärtner, J., Dubrova, E.: Unpacking needs protection: A single-trace secret key recovery attack on dilithium. IACR Communications in Cryptology **1**(3), 1–26 (2024)

Appendix A

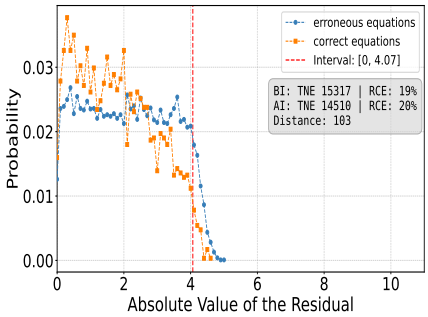


(a) Result after the 1st global iteration

(b) Result after the 2nd global iteration

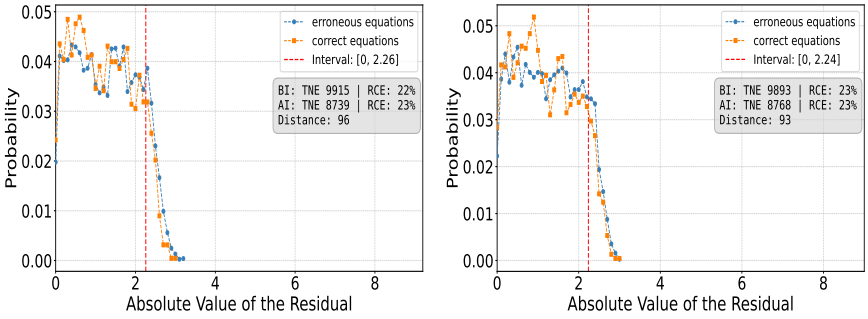


(c) Result after the 3rd global iteration

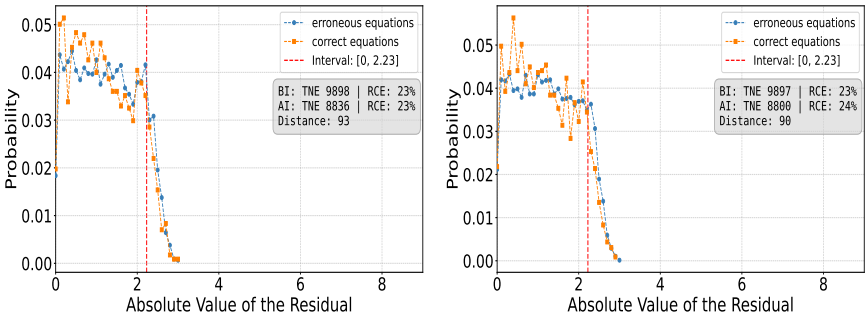


(d) Result after the 4th global iteration

Fig. 9. Performances of least square after different times of Type-I global iteration with equations whose residuals are in the interval of $[0, m_{\mathbf{e}} + 2]$.

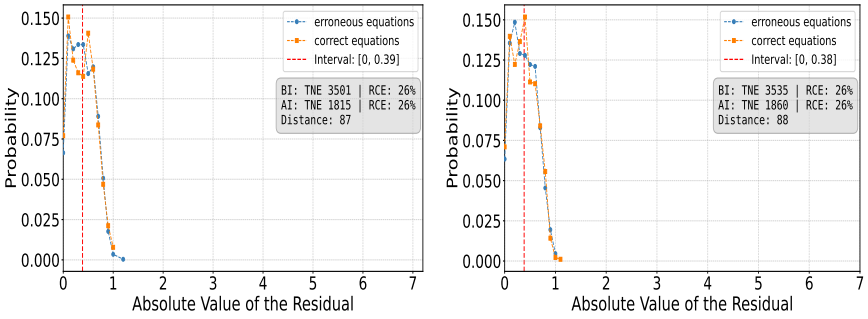


(a) Result after the 1st global iteration (b) Result after the 2nd global iteration



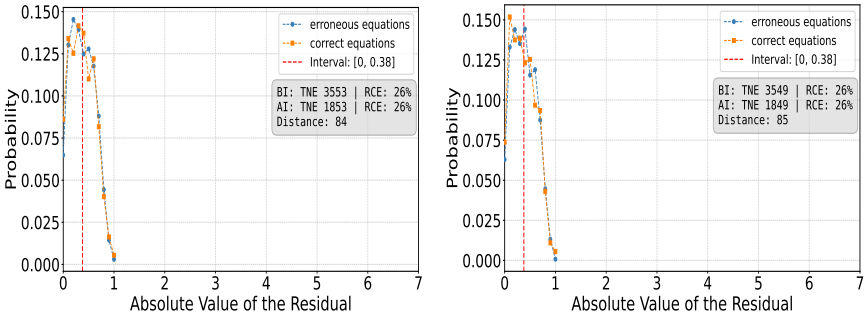
(c) Result after the 3rd global iteration (d) Result after the 4th global iteration

Fig. 10. Performances of least square after different times of Type-I global iteration with equations whose residuals are in the interval of $[0, m_{\mathbf{e}} + 1]$.



(a) Result after the 1st global iteration

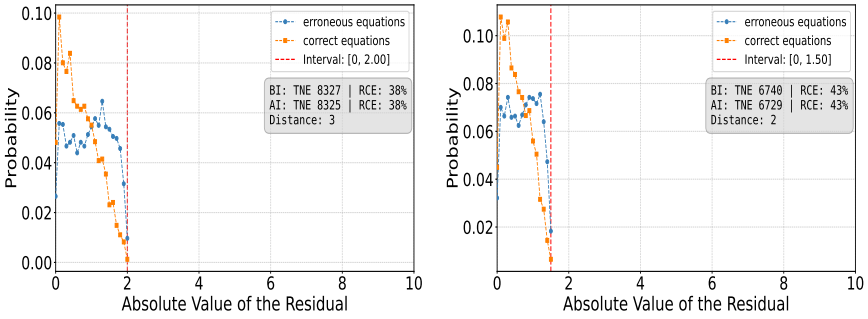
(b) Result after the 2nd global iteration



(c) Result after the 3rd global iteration

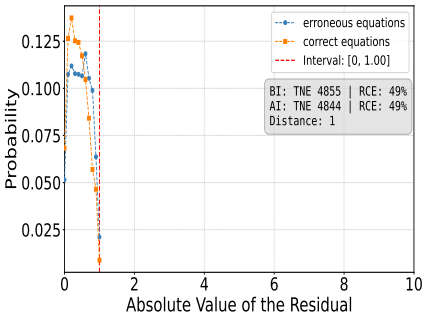
(d) Result after the 4th global iteration

Fig. 11. Performances of least square after different times of Type-I global iteration with equations whose residuals are in the interval of $[0, m_{\mathbf{e}}]$.

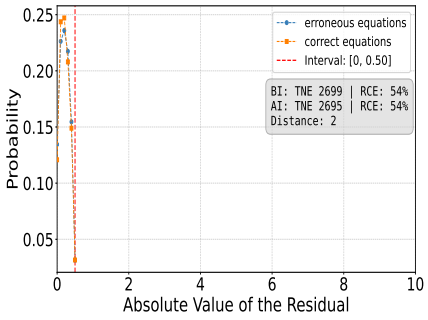


(a) Result of the 6th iteration

(b) Result of the 12th iteration



(c) Result of the 18th iteration



(d) Result of the 24th iteration

Fig. 12. Performances of least square after different times of iteration with equations whose residuals are in the intervals of $[0, 2]$, $[0, 1.5]$, $[0, 1]$ and $[0, 0.5]$.