An Efficient Adaptive Attack Against FESTA

Guoqing Zhou and Maozhi Xu

School of Mathematical Sciences, Peking University, Beijing, China
zgqsms@pku.edu.cn

Abstract. At EUROCRYPT'23, Castryck and Decru, Maino et al., and Robert present efficient attacks against supersingular isogeny Diffie-Hellman key exchange protocol (SIDH). Drawing inspiration from these attacks, Andrea Basso, Luciano Maino, and Giacomo Pope introduce FESTA, an isogeny-based trapdoor function, along with a corresponding IND-CCA secure public key encryption (PKE) protocol at ASIACRYPT'23. FESTA incorporates either a diagonal or circulant matrix into the secret key to mask torsion points.
In this paper, we employ a side-channel attack to construct an auxiliary verification oracle. By querying this oracle, we propose an adaptive attack strategy to recover the secret key in FESTA when the secret matrix is circulant. Compared with existing attacks, our strategy is more efficient and formal. Leveraging these findings, we implement our attack algorithms to recover the circulant matrix in secret key. Finally, we demonstrate that if the secret matrix is circulant, then the adversary can successfully recover FESTA's secret key with a polynomial number of decryption machine queries. Consequently, our paper illustrates that FESTA PKE protocol with secret circulant matrix does not achieve IND-CCA security.

Keywords: Isogeny-based Cryptography · Cryptanalysis · FESTA · Adaptive Attack · Side-channel Attack

1 Introduction

With the rapid advancement of quantum computing, the traditional pubic key cryptosystems are increasingly unable to guarantee digital security [24,26]. To counter the threat posed by quantum computation, post-quantum cryptography has received extensive attention. Isogeny-based cryptography is one of the candidates for post-quantum cryptography. Compared with other post-quantum cryptosystems (e.g., lattice-based [23] and code-based [1]), isogeny-based cryptosystems have the advantage of smaller size of public keys [6], making them more suitable for applications with limited bandwidth (e.g., RS and IoT).

In 2011, Jao and De Feo [14] introduced a supersingular isogeny Diffie-Hellman key exchange protocol (SIDH), relying on isogenies between supersingular elliptic curves. As the endomorphism ring of a supersingular elliptic curve is non-commutative, SIDH is believed to be quantum-resistant [14]. SIDH plays a crucial role in various post-quantum applications, such as PKE scheme [8], signature scheme [13] and key encapsulation protocol SIKE [2].

Prior to 2022, attacks against SIDH were only viable under special scenarios [12,27] and unbalanced parameters [20,21]. However, at EUROCRYPT'23, the underlying hard problem of SIDH has been thoroughly addressed, paving the way for a series of efficient attacks [4,16,22]. The main idea of these attacks is taking advantage of extra torsion points revealed by the participants Alice and Bob and applying Kani's theorem.

Meanwhile, these attacks result in the development of other isogeny-based protocols. Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski [7] introduce a new digital signature scheme SQISignHD inspired by SQISign and Kani's theorem. Compared to classic SQISign, SQISignHD is more efficient and compact. Andrea Basso, Luciano Maino, and Giacomo Pope [3] construct an isogeny-based trapdoor function named FESTA and a corresponding IND-CCA secure PKE protocol. To mask the torsion points, FESTA introduces a secret diagonal matrix or circulant matrix in secret key.

There exist several adaptive attacks against isogeny-based cryptography. A classical one is the attack against SIDH protocol [12], where Galbraith et al. demonstrate its insecurity when the participants use static secret keys. The countermeasures of this attack are relatively expensive. To reduce costs and enable static-static secret keys, Fouotsa and Petit [10] construct a public key validation mechanism and proposed the HealSIDH protocol. Subsequently, Galbraith and Lai [11] point out that the validation mechanism aids in recovering secret keys, and they present an adaptive attack against HealSIDH and corresponding PKE protocol. Against FESTA trapdoor function, Moriya [18] propose a possible adaptive attack under certain assumptions.

It is common to construct an auxiliary oracle in adaptive attacks. The main idea of attack is to substitute honest script with malicious information and deliver the modified script to oracle. Through the output of oracle, the adversary can recover secret information. To attack the FESTA PKE protocol, the oracle should be constructed in actual scenario.

Related work. There are currently two known attacks against FESTA. The first is a polynomial-time attack put forth by Castryck and Vercauteren [5]. Their attack additionally requires that at least one of the basis points in public parameters spans an eigenspace of Frobenius, of an endomorphism of low degree, or of a composition of both. However, the current implementation of FESTA does not choose such a basis.

The second is a possible adaptive attack proposed by Moriya [18]. He construct an auxiliary oracle assuming that the validity of matrix is not checked during the protocol process, but this assumption does not hold in the actual FESTA PKE protocol. Our attack is inspired by Moriya, so we will introduce his work in Sect. 2.4.

Overall, these two attacks prove ineffective against the actual implementation of FESTA. As a consequence, the FESTA PKE protocol is still secure.

Contributions. In this work, we consider a practical adaptive attack against FESTA PKE protocol, and make the following contributions.

(1). We use side-channel attack to distinguish between two exceptions in the decryption algorithm of FESTA PKE protocol. This leads to the construction of an auxiliary verification oracle, aiding in recovering the secret key in protocol.
(2). We introduce a simper method to recover the secret key and prove that FESTA PKE protocol with secret circulant matrix is not IND-CCA secure. Using the attack against SIDH, we find that only a portion of the secret key needs to be recovered for complete key recovery. Additionally, we propose a more efficient and formal adaptive attack designed for secret circulant matrix. In comparison to existing attack, our approach only demands approximately one-eighth of the queries to the decryption machine.
(3). We present our attack algorithms and an implementation in SageMath. Two methods for implementing the verification oracle are available, with the choice of these methods considered as a flag. On a single performance core of an AMD Ryzen 7 7840H CPU, we successfully recover the secret matrix of FESTA with 128-bit security in 2791.565s.

Organization. This paper is organized as follows. In Sect. 2, we introduce preliminary information related to basic knowledge. Section 3 presents an auxiliary verification oracle from side channel. We describe our attack strategy in Sect. 4. Section 5 gives an outline of our implementation. Finally, we conclude this paper in Sect. 6.

2 Preliminaries

In this section, we introduce some mathematical concepts and facts about isogenies and provide a concise overview of necessary isogeny-based cryptosystems.

2.1 Abelian varieties and isogenies

This subsection presents some knowledge about abelian varieties and isogenies, For concrete definitions and rigorous proofs, readers can refer to [25,17,19].

An abelian variety is a complete group variety. For any abelian variety A, there is a unique dual variety $A^\vee$ up to isomorphism. An isogeny between abelian varieties is a surjective homomorphism with finite kernel. The *polarization* of A is the isogeny $\lambda_A = \phi_{\mathcal{L}} : A \to A^\vee$ induced by the ample divisor $\mathcal{L}$. The polarization is *principal* if it is an isomorphism.

The *Weil pairing* on a principally polarized abelian variety (PPAV) is a non-degenerate alternating pairing

$$e_n : A[n] \times A[n] \to \boldsymbol{\mu}_n,$$

where $A[n]$ is the group of all n-torsion points on A and $\boldsymbol{\mu}_n$ is the n-th root group of unity.

Abelian variety A defined over $\mathbb{F}_q$ is denoted by $A/\mathbb{F}_q$. If $A/\mathbb{F}_q$ is a dimension g abelian variety and $\gcd(n,q)=1$, then $A[n]\cong(\mathbb{Z}/n\mathbb{Z})^{2g}$. An *$n$-isogeny* $\phi: A\to B$ between PPAVs is an isogeny such that $\phi^\vee\circ\lambda_B\circ\phi=[n]\circ\lambda_A$, where $\phi^\vee: B^\vee\to A^\vee$ is the dual isogeny and $[n]$ is a scalar multiplication. Denote $\widehat{\phi}=\lambda_A^{-1}\phi^\vee\lambda_B: B\to A$, then $\widehat{\phi}\circ\phi=[n]$. For simplicity, we also call $\widehat{\phi}$ the dual isogeny of ϕ. An n-isogeny between PPAVs defined over $\mathbb{F}_q$ is separable if and only if $\gcd(n,q)=1$. Every separable isogeny between PPAVs can be characterized by its kernel up to isomorphism. If ϕ is a separable n-isogeny from dimension g PPAV A, then $\ker\phi$ is a maximal isotropic subgroup of $A[n]$ with respect to Weil pairing e_n. It holds that

$$\ker\phi\cong\ker\widehat{\phi}\cong\prod_{i=1}^{g}(\mathbb{Z}/n_i\mathbb{Z}\times\mathbb{Z}/\frac{n}{n_i}\mathbb{Z}),$$

where $n_i\mid n,\ i=1,2,\cdots,g$. It follows that $\#\ker\phi=n^g$.

PPAVs of dimension one are just elliptic curves. Let $E/\mathbb{F}_{p^k}$ be an elliptic curve. If $E[p]=\{0\}$, then E is a *supersingular* elliptic curve. The endomorphism ring of supersingular elliptic curve is non-commutative, which makes isogeny-based cryptography resistant to quantum attacks. In the case of dimension two, every principally polarized abelian surface (PPAS) is isomorphic to the product of two elliptic curves or the jacobian of a hyperelliptic curve of genus two.

2.2 Polynomial-time attack against SIDH

SIDH is a well-known key exchange protocol proposed by Jao and De Feo in [14]. It has been extensively studied over the past decade [12,27,20,21]. But SIDH protocol is completely broken in polynomial time at EUROCRYPT'23 [4,16,22].

SIDH protocol uses isogenies between supersingular elliptic curves, and it can be summarized as follows. In set-up, we select prime $p=2^a3^bf-1$ where $2^a\approx3^b$ and f is a small factor, select a supersingular elliptic curve $E_0/\mathbb{F}_{p^2}$, and generate the basis $\{P_1,Q_1\}$ of $E_0[2^a]$ and the basis $\{P_2,Q_2\}$ of $E_0[3^b]$. Then Alice and Bob perform the key exchange as shown in Figure 1. The red lines are computed by Alice and the blue lines are computed by Bob.

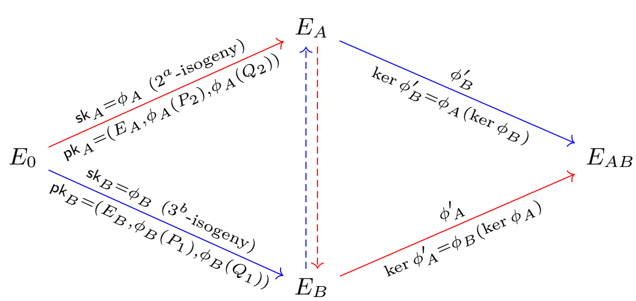

Fig. 1. SIDH protocol.

In order for Alice to generate ϕ'_A and Bob to generate ϕ'_B, the public keys contain four extra torsion points. It is a leakage of secret information. All the attacks in [4,16,22] are based on following Kani's theorem[15, Theorem 2.3].

Theorem 1 (Kani's theorem). *Suppose that E_0 is an elliptic curve, $\phi_B : E_0 \to E_B$ is an N_B-isogeny, and $\gamma : E_0 \to E_C$ is an $(N_A - N_B)$-isogeny, where N_A and N_B are coprime. There is a commutative diagram*

$$\begin{array}{ccc} E_0 & \xrightarrow{\phi_B} & E_B \\ {\scriptstyle\gamma}\downarrow & & \downarrow{\scriptstyle\gamma'} \\ E_C & \xrightarrow{\phi'_B} & E_{BC}. \end{array}$$

Then the following map is an N_A-isogeny between PPAS:

$$F : E_B \times E_C \to E_0 \times E_{BC}$$
$$\begin{pmatrix} R \\ S \end{pmatrix} \mapsto \begin{pmatrix} \widehat{\phi_B}(R) + \widehat{\gamma}(S) \\ \gamma'(R) - \phi'_B(S) \end{pmatrix}.$$

The kernel of the isogeny is $\ker F = \langle\ (\phi_B(P), \gamma(P)), (\phi_B(Q), \gamma(Q))\rangle$, *where* $\{P, Q\}$ *is a basis of* $E_0[N_A]$.

Without loss of generality, suppose that $2^a > 3^b$ in the SIDH protocol. Attacker possesses knowledge of the action of ϕ_B on $E_0[N_A]$ from pk_B. If the attacker has ability to construct a $(2^a - 3^b)$-isogeny γ, then Kani's theorem reveals that through its kernel attacker can generate an isogeny F between PPAS, where the isogeny $\widehat{\phi_B}$ is a component. Once recovering the isogeny $\widehat{\phi_B}$, the secret kernel $\ker \phi_B = \widehat{\phi_B}(E_B[N_B])$ can be computed directly. Robert [22] improved this method, eliminating the requirement that $N_A > N_B$ and making it sufficient that $N_A^2 > N_B$.

Therefore, SIDH attacks can be abstracted as a generic algorithm that recovers an isogeny $\phi : E_0 \to E_1$ of degree d when it receives the curve E_0, E_1, the degree d, a basis $\{P_0, Q_0\}$ of $E_0[n]$ where $n^2 \geqslant d$, and points $\{P_1 = \phi(P_0), P_2 = \phi(Q_0)\}$. We denote this algorithm as

$$\mathsf{TorAtk}(E_0, P_0, Q_0, E_1, P_1, Q_1, d).$$

If the input to TorAtk is invalid, then the algorithm is configured to output $\perp$.

2.3 FESTA PKE protocol

This subsection introduces an overview of FESTA PKE protocol, which is the target of our cryptanalysis. Based on the polynomial-time attack against SIDH mentioned in Section 2.2, Basso et al. [3] propose a FESTA trapdoor function and obtain a FESTA PKE protocol using the OAEP transform [9].

Now we introduce the construction of FESTA trapdoor function.

- **Public parameters.** Let d_1, d_2, d_A be odd integers such that they are pairwise coprime. Let m_1, m_2 be integers such that $m_1^2 + m_2^2 d_1 d_2 d_A = 2^b$. Define a prime $p = 2^b d_1 d_2 d_A f - 1$ where f is a small integer. Let $E_0/\mathbb{F}_{p^2}$ be a supersingular elliptic curve with $j(E_0) \neq 0, 1728$. Let $\{P_b, Q_b\}$ be a basis of $E_0[2^b]$. Define $\mathcal{M}_b$ as a commutative subgroup of $\mathrm{GL}(2, \mathbb{Z}/2^b\mathbb{Z})$.
- **Key generation.** Compute a d_A-isogeny $\phi_A : E_0 \to E_A$. Take a random matrix $\boldsymbol{A} \in \mathcal{M}_b$ and compute

$$\begin{pmatrix} R_A \\ S_A \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} \phi_A(P_b) \\ \phi_A(Q_b) \end{pmatrix}.$$

  Finally, set (E_A, R_A, S_A) as public key and keep $(\phi_A, \boldsymbol{A})$ as secret key.
- **FESTA trapdoor function.** Input a subgroup $\langle K_1 \rangle \subseteq E_0[d_1]$ of order d_1, a subgroup $\langle K_2 \rangle \subseteq E_A[d_2]$ of order d_2, and a matrix $\boldsymbol{B} \in \mathcal{M}_b$. Compute isogenies $\phi_1 : E_0 \to E_0/\langle K_1 \rangle = E_1$, $\phi_2 : E_A \to E_A/\langle K_2 \rangle = E_2$, and

$$\begin{pmatrix} R_1 \\ S_1 \end{pmatrix} = \boldsymbol{B} \begin{pmatrix} \phi_1(P_b) \\ \phi_1(Q_b) \end{pmatrix}, \quad \begin{pmatrix} R_2 \\ S_2 \end{pmatrix} = \boldsymbol{B} \begin{pmatrix} \phi_2(R_A) \\ \phi_2(S_A) \end{pmatrix}.$$

  Output $(E_1, (R_1, S_1), E_2, (R_2, S_2))$, i.e.,

$$f_{(E_A, R_A, S_A)}(\langle K_1 \rangle, \langle K_2 \rangle, \boldsymbol{B}) = (E_1, (R_1, S_1), E_2, (R_2, S_2)).$$

- **Inverse function.** Input a tuple $(E_1, (R_1, S_1), E_2, (R_2, S_2))$. Using matrix $\boldsymbol{A}$ in secret key, compute

$$\begin{pmatrix} R_2' \\ S_2' \end{pmatrix} = d_1 \boldsymbol{A}^{-1} \begin{pmatrix} R_2 \\ S_2 \end{pmatrix}.$$

  Compute the isogeny $\psi = \phi_2 \circ \phi_A \circ \widehat{\phi_1} : E_1 \to E_2$ through $\mathsf{TorAtk}(E_1, R_1, S_1, E_2, R_2', S_2', d_1 d_A d_2)$. Recover the kernel $\langle K_1 \rangle$ of isogeny ϕ_1 and the kernel $\langle K_2 \rangle$ of isogeny ϕ_2 from ψ using ϕ_A in secret key. Compute $\boldsymbol{B} \in \mathcal{M}_b$ such that

$$\begin{pmatrix} R_1 \\ S_1 \end{pmatrix} = \boldsymbol{B} \begin{pmatrix} \phi_1(P_b) \\ \phi_1(Q_b) \end{pmatrix}.$$

  Output $(\langle K_1 \rangle, \langle K_2 \rangle, \boldsymbol{B})$, i.e.,

$$f^{-1}_{(\phi_A, \boldsymbol{A})}(E_1, (R_1, S_1), E_2, (R_2, S_2)) = (\langle K_1 \rangle, \langle K_2 \rangle, \boldsymbol{B}).$$

Anyone with knowledge of the public key can compute the trapdoor function, but only the individual with the secret key possesses the capability to compute the inverse function. The construction of FESTA trapdoor function can be summarized in Fig. 2.

$$\begin{tikzcd} E_0 \arrow[r, "\phi_A"] \arrow[d, "\phi_1"'] & E_A \arrow[d, "\phi_2"] \\ E_1 \arrow[r, dashed, "\psi"] & E_2 \end{tikzcd}$$

$$\begin{pmatrix} P_b \\ Q_b \end{pmatrix} \qquad \begin{pmatrix} R_A \\ S_A \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} \phi_A(P_b) \\ \phi_A(Q_b) \end{pmatrix}$$

$$\begin{pmatrix} R_1 \\ S_1 \end{pmatrix} = \boldsymbol{B} \begin{pmatrix} \phi_1(P_b) \\ \phi_1(Q_b) \end{pmatrix} \qquad \begin{pmatrix} R_2 \\ S_2 \end{pmatrix} = \boldsymbol{B} \begin{pmatrix} \phi_2(R_A) \\ \phi_2(S_A) \end{pmatrix}$$

Fig. 2. The FESTA trapdoor function

Using the OAEP transform on FESTA trapdoor function, we obtain FESTA PKE protocol. Given two random oracles $G : \mathbb{Z}/d_2\mathbb{Z} \times \mathcal{M}_b \to \mathbb{Z}/d_1\mathbb{Z}$ and $H : \mathbb{Z}/d_1\mathbb{Z} \to \mathbb{Z}/d_2\mathbb{Z} \times \mathcal{M}_b$. The encrpytion algorithm and decryption algorithm are outlined as follows.

- **Encryption.** Bob chooses a plaintext $m \in \mathbb{Z}/d_1\mathbb{Z}$. He randomly chooses $r \in \mathbb{Z}/d_2\mathbb{Z}$ and $R \in \mathcal{M}_b$ and computes $s = m + G(r, R)$, $(x, X) = H(s)$, $t = x + r$, $T = XR$. Then, he generates the points $K_1 = P_1 + [s]Q_1$ and $K_2 = P_2 + [t]Q_2$, where $\{P_i, Q_i\}$ is the canonical basis of $E_i[d_i]$, $i = 1, 2$. The ciphertext $\mathsf{ct} = f_{(E_A,R_A,S_A)}(\langle K_1\rangle, \langle K_2\rangle, T)$
- **Decryption.** Receiving the cipertext ct, Alice computes $(s, t, T) = f^{-1}_{(\phi_A,\boldsymbol{A})}(\mathsf{ct})$. She computes $(x, X) = H(s)$, $r = t-x$, $R = X^{-1}T$. Then, she gets the plaintext $m = s - G(r, R)$.

Basso et al. claim that FESTA PKE protocol is IND-CCA secure. That is to say, given two messages, any probabilistic polynomial-time adversary cannot distinguish which message has been encrypted even if they can ask to decrypt some ciphertexts different from the challenge ciphertext at any point during the attack.

Remark 1. To generate suitable parameter sets in concrete instantiation, the secret isogeny ϕ_A is split as a composition of two isogenies: $\phi_A : E_0 \xrightarrow{\phi_{A,1}} \tilde{E_A} \xrightarrow{\phi_{A,2}} E_A$, where the degrees of of $\phi_{A,1}$ and $\phi_{A,2}$ are $d_{A,1}$ and $d_{A,2}$ respectively. The actual parameters satisfy $m_1^2 d_{A,1} d_1 + m_2^2 d_{A,2} d_2 = 2^b$. The inverse function in concrete instantiation is a variant of that described above, but it doesn't affect our attack in Section 3 and Section 4.

2.4 Possible adaptive attack against FESTA trapdoor function

Moriya [18] shows that an adaptive attack can be considered if the FESTA trapdoor function was used in the wrong way. He presents a possible adaptive attack against FESTA trapdoor function under the following assumption:

1. The adversary has access to a decryption machine.
2. The recipient does not check matrix $\boldsymbol{B} \in \mathcal{M}_b$ in the decryption process.

In FESTA trapdoor function, there is a relationship between torsion points (R_1, S_1) and (R_2, S_2):

$$\begin{aligned}\begin{pmatrix} R_2 \\ S_2 \end{pmatrix} &= \boldsymbol{B} \cdot \phi_2 \circ \boldsymbol{A} \cdot \phi_A \begin{pmatrix} P_b \\ Q_b \end{pmatrix} \\ &= \boldsymbol{B}\boldsymbol{A} \cdot \phi_2 \circ \phi_A \circ \frac{1}{d_1} \circ \widehat{\phi_1} \circ \boldsymbol{B}^{-1} \begin{pmatrix} R_1 \\ S_1 \end{pmatrix} \\ &= \boldsymbol{A} \cdot \frac{1}{d_1} \phi_2 \circ \phi_A \circ \widehat{\phi_1} \begin{pmatrix} R_1 \\ S_1 \end{pmatrix}. \end{aligned} \tag{1}$$

This relationship serves as a proof of the correctness of the inverse function, and it also means that as long as the torsion points satisfy equation (1), then the inverse function will compute correct isogeny $\psi = \phi_2 \circ \phi_A \circ \widehat{\phi_1}$ and output the kernel groups of isogenies ϕ_1 and ϕ_2. Dishonest torsion points can lead to an invalid matrix $\boldsymbol{B}$. Assuming that the validity of matrix is not checked during the protocol process, the decryption machine will determine whether equation (1) holds. Hence, Moriya introduces an auxiliary verification oracle as follows:

$$O(E_1, (R_1, S_1), E_2, (R_2, S_2)) = \begin{cases} 1\ , \text{ if } \begin{pmatrix} R_2 \\ S_2 \end{pmatrix} = \boldsymbol{A} \cdot \frac{1}{d_1} \phi_2 \circ \phi_A \circ \widehat{\phi_1} \begin{pmatrix} R_1 \\ S_1 \end{pmatrix}, \\ 0\ , \text{ otherwise.} \end{cases}$$

Another observation is that if the secret matrix $\boldsymbol{A}$ is leaked, then the adversary can compute

$$\begin{pmatrix} \phi_A(P_b) \\ \phi_A(Q_b) \end{pmatrix} = \boldsymbol{A}^{-1} \begin{pmatrix} R_A \\ S_A \end{pmatrix}$$

and recover the secret isogeny ϕ_A through the algorithm $\mathsf{TorAtk}($ E_0, P_b, Q_b, E_A, $\phi_A(P_b)$, $\phi_A(Q_b)$, $d_A)$. Therefore, the adversary only needs to recover matrix $\boldsymbol{A}$ to get the complete secret key.

In the concrete instantiation of FESTA, $\mathcal{M}_b$ is the diagonal matrix group or circulant matrix group. If $\mathcal{M}_b$ represents circulant matrix group, then Moriya gives a strategy to recover secret matrix $\boldsymbol{A}$ in at most $8b - 1$ queries to the verification oracle O. The main method of Moriya's attack is to replace the honest ciphertext with malicious torsion points and deliver the script to oracle O. Through the output of oracle, the information of secret matrix $\boldsymbol{A}$ can be recovered.

But unfortunately, Moriya's second assumption does not hold in actual FESTA PKE protocol. To be specific, if the the matrix $\boldsymbol{B} \notin \mathcal{M}_b$, then the inverse function algorithm [3, Algorithm 7] will return $\perp$. Hence, even though the torsion points (R_1, S_1) and (R_2, S_2) satisfy equation (1), there is no output in the inverse function. It means that verification oracle O doesn't work in the actual attack scenario. Therefore, FESTA PKE protocol is still secure under Moriya's adaptive attack.

3 Revalidated Oracle From Side Channel

In this section, we show that it is feasible to revalidate the verification oracle O proposed by Moriya via side channel. This revalidation renders Moriya's attack and our simpler attack effective against actual FESTA PKE protocol.

First, we study the check process of matrix $\boldsymbol{B}$ in the FESTA inverse function. Suppose that we have a honest script $(E_1, (R_1, S_1), E_2, (R_2, S_2))$ and choose malicious torsion points

$$\begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} = \boldsymbol{M} \begin{pmatrix} R_1 \\ S_1 \end{pmatrix} \ , \ \begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} = \boldsymbol{N} \begin{pmatrix} R_2 \\ S_2 \end{pmatrix} .$$

If $\begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} = \boldsymbol{A} \cdot \frac{1}{d_1} \phi_2 \circ \phi_A \circ \widehat{\phi_1} \begin{pmatrix} P_1 \\ Q_1 \end{pmatrix}$, then honest isogenies ϕ_1 and ϕ_2 will be generated in the inverse function, and recipient will compute $\boldsymbol{B'}$ such that $\begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} = \boldsymbol{B'} \begin{pmatrix} \phi_1(P_b) \\ \phi_1(Q_b) \end{pmatrix}$. We know that $\begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} = \boldsymbol{M} \begin{pmatrix} R_1 \\ S_1 \end{pmatrix} = \boldsymbol{MB} \begin{pmatrix} \phi_1(P_b) \\ \phi_1(Q_b) \end{pmatrix}$, so recipient will get matrix $\boldsymbol{B'} = \boldsymbol{MB}$. $\mathcal{M}_b$ is a group and $\boldsymbol{B} \in \mathcal{M}_b$, so $\boldsymbol{B'} \in \mathcal{M}_b$ if and only if $\boldsymbol{M} \in \mathcal{M}_b$. Therefore, the check process of matrix $\boldsymbol{B}$ passes if and only if malicious matrix $\boldsymbol{M} \in \mathcal{M}_b$.

It is hard to attack FESTA if we require malicious matrix $\boldsymbol{M} \in \mathcal{M}_b$, so we need to skip this check of matrix $\boldsymbol{B}$ in some sense. We find that in the concrete implementation of FESTA decryption algorithm,

1. the program will throw a 'ValueError' if the malicious torsion points satisfy

$$\begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} \neq \boldsymbol{A} \cdot \frac{1}{d_1} \phi_2 \circ \phi_A \circ \widehat{\phi_1} \begin{pmatrix} P_1 \\ Q_1 \end{pmatrix},$$

2. and the program will return 'False' if the malicious torsion points satisfy

$$\begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} = \boldsymbol{A} \cdot \frac{1}{d_1} \phi_2 \circ \phi_A \circ \widehat{\phi_1} \begin{pmatrix} P_1 \\ Q_1 \end{pmatrix}, \text{ but } \boldsymbol{M} \notin \mathcal{M}_b.$$

Therefore, given a FESTA decryption machine, we can distinguish between these two exceptions through catching 'ValueError'. It means that we revalidate the verification oracle O. This gives a straightforward implementation of oracle O as Algorithm 1.

Algorithm 1: Straightforward implementation of verification oracle O

Input: A well-formatted script $(E_1, (P_1, Q_1), E_2, (P_2, Q_2))$, and FESTA decryption machine Dec.

Output: 1 or 0.

if $\mathsf{Dec}(E_1, (P_1, Q_1), E_2, (P_2, Q_2))$ *throws a 'ValueError'* **then return** 0;

else return 1;

Note that there are many steps in decryption algorithm [3, Algorithm 7] between the above two exceptions. Even though the program returns same symbol in the above two exceptions, we can still infer the output of oracle through the running time in decryption process. For instance, on a single performance core of an AMD Ryzen 7 7840H CPU, the average running time in decryption process of exception 1 is 6.682s, while that of exception 2 is 10.474s. It is feasible to distinguish between these two exceptions. This gives a time-based implementation of oracle O as outlined in Algorithm 2.

Algorithm 2: Time-based implementation of verification oracle O

Input: A well-formatted script $(E_1, (P_1, Q_1), E_2, (P_2, Q_2))$, an honest ciphertext $(E_1, (R_1, S_1), E_2, (R_2, S_2))$, and FESTA decryption machine Dec.

Output: 1 or 0.

Record the average running time T_1 of

$$\mathsf{Dec}(E_1, (R_1 + [2^{b-1}]S_1, S_1), E_2, (R_2 + [2^{b-1}]S_2, S_2)); \tag{2}$$

Record the average running time T_2 of

$$\mathsf{Dec}(E_1, (R_1, S_1 + [2^{b-1}]R_1), E_2, (R_2 + [2^{b-1}]S_2, S_2)); \tag{3}$$

Compute $T = (T_1 + T_2)/2$;

if *the running time of* $\mathsf{Dec}(E_1, (P_1, Q_1), E_2, (P_2, Q_2))$ *is smaller than* T **then**

  return 0

else return 1;

Remark 2. The torsion points in formula (2) and formula (3) lead to different exceptions, which will be explained in Section 4.2.

4 A More Efficient Strategy for Circulant Matrix

In this section, we present a more efficient and formal strategy to recover the secret circulant matrix. Our strategy only needs at most $b - 2$ queries to the oracle O.

4.1 Main idea

The first idea is that only half of the bits of secret matrix $\boldsymbol{A}$ is sufficient for attack. From the public parameters $m_1^2 + m_2^2 d_1 d_A d_2 = 2^b$, we know that $2^b \geq d_A$, i.e., $(2^{\frac{b}{2}})^2 \geq d_A$. Using TorAtk algorithm, we only need to recover the action of isogeny on $2^{\frac{b}{2}}$-torsion basis. In key-generation of FESTA, $\begin{pmatrix} R_A \\ S_A \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} \phi_A(P_b) \\ \phi_A(Q_b) \end{pmatrix}$

are the torsion points in public key. Thus,

$$\begin{pmatrix}[2^{\frac{b}{2}}]R_A\\ [2^{\frac{b}{2}}]S_A\end{pmatrix} = \boldsymbol{A}\begin{pmatrix}\phi_A([2^{\frac{b}{2}}]P_b)\\ \phi_A([2^{\frac{b}{2}}]Q_b)\end{pmatrix} = \boldsymbol{A}_1\begin{pmatrix}\phi_A([2^{\frac{b}{2}}]P_b)\\ \phi_A([2^{\frac{b}{2}}]Q_b)\end{pmatrix},$$

where $\{[2^{\frac{b}{2}}]P_b,\ [2^{\frac{b}{2}}]Q_b\}$ is a basis of $E_0[2^{\frac{b}{2}}]$ and $\boldsymbol{A}_1 \equiv \boldsymbol{A} \bmod 2^{\frac{b}{2}}$. If we recover matrix $\boldsymbol{A}_1$, then we can directly compute

$$\begin{pmatrix}\phi_A([2^{\frac{b}{2}}]P_b)\\ \phi_A([2^{\frac{b}{2}}]Q_b)\end{pmatrix} = \boldsymbol{A}_1^{-1}\begin{pmatrix}[2^{\frac{b}{2}}]R_A\\ [2^{\frac{b}{2}}]S_A\end{pmatrix}$$

and recover ϕ_A from $\mathsf{TorAtk}(E_0, [2^{\frac{b}{2}}]P_b, [2^{\frac{b}{2}}]Q_b, E_A, \phi_A([2^{\frac{b}{2}}]P_b), \phi_A([2^{\frac{b}{2}}]Q_b), d_A)$. So the matrix $\boldsymbol{A}_1$ is sufficient for attack. In contrast to Moriya's attack, only half of the bits of matrix $\boldsymbol{A}$ need to be recovered.

The second idea is that malicious torsion points can be represented by matrix formally. Suppose we get a honest receipt $(E_1, (R_1, S_1), E_2, (R_2, S_2))$, and we write malicious points

$$\begin{pmatrix}P_1\\ Q_1\end{pmatrix} = \boldsymbol{M}\begin{pmatrix}R_1\\ S_1\end{pmatrix}\ ,\ \begin{pmatrix}P_2\\ Q_2\end{pmatrix} = \boldsymbol{N}\begin{pmatrix}R_2\\ S_2\end{pmatrix},$$

where matrices $\boldsymbol{M}, \boldsymbol{N} \in \mathrm{GL}(2, \mathbb{Z}/2^b\mathbb{Z})$ represent our malicious choice. Then

$$\begin{aligned}\begin{pmatrix}P_2\\ Q_2\end{pmatrix} &= \boldsymbol{N}\begin{pmatrix}R_2\\ S_2\end{pmatrix}\\ &= \boldsymbol{NA}\cdot\frac{1}{d_1}\phi_2\circ\phi_A\circ\widehat{\phi_1}\begin{pmatrix}R_1\\ S_1\end{pmatrix} && (\triangleright \text{ formula } (1))\\ &= \boldsymbol{NA}\cdot\frac{1}{d_1}\phi_2\circ\phi_A\circ\widehat{\phi_1}\ \boldsymbol{M}^{-1}\cdot\begin{pmatrix}P_1\\ Q_1\end{pmatrix}\\ &= \boldsymbol{NAM}^{-1}\cdot\frac{1}{d_1}\phi_2\circ\phi_A\circ\widehat{\phi_1}\begin{pmatrix}P_1\\ Q_1\end{pmatrix}.\end{aligned}$$

Hence, $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1$ if and only if $\boldsymbol{NAM}^{-1} = \boldsymbol{A}$. By selecting appropriate malicious matrices $\boldsymbol{M}$ and $\boldsymbol{N}$, the equality $\boldsymbol{NAM}^{-1} = \boldsymbol{A}$ can reveal information about secret matrix $\boldsymbol{A}$.

Suppose the secret circulant matrix $\boldsymbol{A} = \begin{pmatrix}\gamma & \delta\\ \delta & \gamma\end{pmatrix}$, where $\det \boldsymbol{A} = \gamma^2 - \delta^2 \in (\mathbb{Z}/2^b\mathbb{Z})^\times$. It follows that the parity of γ and δ is opposite. To recover matrix $\boldsymbol{A}_1 \equiv \boldsymbol{A} \bmod 2^{\frac{b}{2}}$, we need to recover the first $\frac{b}{2}$ bits of γ and δ.

4.2 The first bits of γ and δ

In this subsection, we recover the fisrt bits of γ and δ. We choose malicious matrices $\boldsymbol{M}_0 = \boldsymbol{N}_0 = \begin{pmatrix}1 & 2^{b-1}\\ 0 & 1\end{pmatrix}$, then

$$\boldsymbol{N}_0\boldsymbol{A}\boldsymbol{M}_0^{-1} = \begin{pmatrix}1 & 2^{b-1}\\ 0 & 1\end{pmatrix}\begin{pmatrix}\gamma & \delta\\ \delta & \gamma\end{pmatrix}\begin{pmatrix}1 & 2^{b-1}\\ 0 & 1\end{pmatrix} = \begin{pmatrix}\gamma + 2^{b-1}\delta & \delta\\ \delta & \gamma + 2^{b-1}\delta\end{pmatrix}.$$

Thus, $\boldsymbol{N}_0\boldsymbol{A}\boldsymbol{M}_0^{-1} = \boldsymbol{A}$ if and only if $2^{b-1}\delta = 0$ in $\mathbb{Z}/2^b\mathbb{Z}$, equivalently, δ is even. So we choose

$$\begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} = \boldsymbol{M}_0 \begin{pmatrix} R_1 \\ S_1 \end{pmatrix}, \quad \begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} = \boldsymbol{N}_0 \begin{pmatrix} R_2 \\ S_2 \end{pmatrix},$$

then $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1$ if and only if δ is even. That is to say, after querying the verification oracle once, we can recover the first bits of γ and δ. We summarize this step in Algorithm 3.

Algorithm 3: The fisrt bits of secret matrix

Input: The public parameter and public key $(p, E_0, P_b, Q_b, E_A, R_A, S_A)$, an honest ciphertext $(E_1, (R_1, S_1), E_2, (R_2, S_2))$, and the verification oracle O.

Output: The first bits of γ and δ.

Set $\begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 2^{b-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_1 \\ S_1 \end{pmatrix}$, $\begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 2^{b-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_2 \\ S_2 \end{pmatrix}$;

if $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1$ **then** set $\gamma \leftarrow 1,\ \delta \leftarrow 0$;

else set $\gamma \leftarrow 0,\ \delta \leftarrow 1$;

return $\gamma,\ \delta$;

Note that if we choose $\boldsymbol{M}_0 = \begin{pmatrix} 1 & 0 \\ 2^{b-1} & 1 \end{pmatrix}$ and $\boldsymbol{N}_0 = \begin{pmatrix} 1 & 2^{b-1} \\ 0 & 1 \end{pmatrix}$, then

$$\boldsymbol{N}_0\boldsymbol{A}\boldsymbol{M}_0^{-1} = \begin{pmatrix} 1 & 2^{b-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \gamma & \delta \\ \delta & \gamma \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 2^{b-1} & 1 \end{pmatrix} = \begin{pmatrix} \gamma & \delta + 2^{b-1}\gamma \\ \delta + 2^{b-1}\gamma & \gamma \end{pmatrix}.$$

Thus, $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1 \Leftrightarrow \boldsymbol{N}_0\boldsymbol{A}\boldsymbol{M}_0^{-1} = \boldsymbol{A} \Leftrightarrow \gamma$ is even. Exploiting the fact that only one of γ and δ is even, these two selections of torsion points will lead to distinct outputs in verification oracle O. It follows that formula (2) and formula (3) in Algorithm 2 correspond to different exceptions.

4.3 The other bits of γ and δ

Now we can recover the other bits of γ and δ by induction. Similar to the GPST attack [12], we write $\gamma = \gamma_i + a_i \cdot 2^i + \gamma'$ and $\delta = \delta_i + b_i \cdot 2^i + \delta'$, where γ_i and δ_i are known and $2^{i+1} \mid \gamma'$, $2^{i+1} \mid \delta'$. To recover a_i and b_i in every iteration, we need the following two lemmas.

Lemma 1. *Let $\gamma, \delta \in \mathbb{Z}/2^b\mathbb{Z}$. $\xi = \gamma^2 - \delta^2$. Write $\gamma = \gamma_i + a_i \cdot 2^i + \gamma'$ and $\delta = \delta_i + b_i \cdot 2^i + \delta'$, where $i \leqslant b-1$. It holds that when $2 \leqslant i \leqslant b-2$,*

$$\xi - \gamma_i^2 + \delta_i^2 \equiv \begin{cases} 2^{i+1} \cdot a_i \pmod{2^{i+2}}, & \text{if } \gamma \text{ is odd and } \delta \text{ is even,} \\ 2^{i+1} \cdot b_i \pmod{2^{i+2}}, & \text{if } \gamma \text{ is even and } \delta \text{ is odd.} \end{cases}$$

Proof. When $i \geqslant 2$, $(2^i)^2 \equiv 0 \bmod 2^{i+2}$, so

$$\begin{aligned}\xi &\equiv \gamma_{i+2}^2 - \delta_{i+2}^2 \bmod 2^{i+2} \\ &= (\gamma_i + a_i 2^i + a_{i+1}2^{i+1})^2 - (\delta_i + b_i 2^i + b_{i+1}2^{i+1})^2 \\ &\equiv (\gamma_i^2 + a_i\gamma_i 2^{i+1}) - (\delta_i^2 + b_i\delta_i 2^{i+1}) \bmod 2^{i+2} \\ &\equiv \gamma_i^2 - \delta_i^2 + a_i\gamma_i \cdot 2^{i+1} + b_i\delta_i \cdot 2^{i+1} \bmod 2^{i+2}.\end{aligned}$$

If γ is odd and δ is even, then γ_i is odd and δ_i is even,

$$\xi - \gamma_i^2 + \delta_i^2 \equiv 2^{i+1} \cdot a_i \pmod{2^{i+2}}.$$

If γ is even and δ is odd, then similarly $\xi - \gamma_i^2 + \delta_i^2 \equiv 2^{i+1} \cdot b_i \pmod{2^{i+2}}$. □

Lemma 2. *Let $\gamma, \delta \in \mathbb{Z}/2^b\mathbb{Z}$. Write $\gamma = \gamma_i + a_i \cdot 2^i + \gamma'$ and $\delta = \delta_i + b_i \cdot 2^i + \delta'$, where $i \leqslant b-1$. It holds that in $\mathbb{Z}/2^b\mathbb{Z}$,*

$$-2^{b-i-1}\delta_i \cdot \gamma + 2^{b-i-1}\gamma_i \cdot \delta = \begin{cases} 2^{b-1} \cdot b_i \ , & \text{if } \gamma \text{ is odd and } \delta \text{ is even,} \\ 2^{b-1} \cdot a_i \ , & \text{if } \gamma \text{ is even and } \delta \text{ is odd.} \end{cases}$$

Proof. In $\mathbb{Z}/2^b\mathbb{Z}$, it holds that

$$\begin{aligned}-2^{b-i-1}\delta_i \cdot \gamma + 2^{b-i-1}\gamma_i \cdot \delta &= 2^{b-i-1}[-\delta_i \cdot (\gamma_i + a_i \cdot 2^i + \gamma') + \gamma_i \cdot (\delta_i + b_i \cdot 2^i + \delta')] \\ &= 2^{b-i-1}(-\delta_i \cdot a_i \cdot 2^i + \gamma_i \cdot b_i \cdot 2^i) \\ &= 2^{b-1}(b_i\gamma_i - a_i\delta_i).\end{aligned}$$

If γ is odd and δ is even, then γ_i is odd and δ_i is even, $2^{b-1}(b_i\gamma_i - a_i\delta_i) = 2^{b-1} \cdot b_i$. If γ is even and δ is odd, then similarly $2^{b-1}(b_i\gamma_i - a_i\delta_i) = 2^{b-1} \cdot a_i$. □

For simplicity, we write $T_i = -2^{b-i-1}\delta_i \cdot \gamma + 2^{b-i-1}\gamma_i \cdot \delta$. It will be used in subsequent steps.

Case 1: $\boldsymbol{\gamma}$ is odd and $\boldsymbol{\delta}$ is even. In this case, γ_i is odd and δ_i is even for every $i \leqslant b-1$. The public keys of FESTA trapdoor function are defined by following set:

$$\mathcal{A}^{\mathsf{pk}} = \left\{ (E_A, R_A, S_A) \;\middle|\; \begin{array}{l} \phi_A : E_0 \to E_A, \deg(\phi_A) = d_A, \\ \boldsymbol{A} \in \mathcal{M}_b, \begin{pmatrix} R_A \\ S_A \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} \phi_A(P_b) \\ \phi_A(Q_b) \end{pmatrix} \end{array} \right\}.$$

Through Weil pairing

$$e_{2^b}(R_A, S_A) = e_{2^b}(\phi_A(P_b), \phi_A(Q_b))^{\det \boldsymbol{A}} = e_{2^b}(P_b, Q_b)^{d_A \cdot \det \boldsymbol{A}},$$

we can solve a discrete logarithm to get $\det \boldsymbol{A} = \xi \in \mathbb{Z}/2^b\mathbb{Z}$. When $2 \leqslant i \leqslant b-2$, from Lemma 1, we know that $\xi - \gamma_i^2 + \delta_i^2 \equiv 2^{i+1} \cdot a_i \pmod{2^{i+2}}$. γ_i and δ_i are known, which follows that we can recover a_i directly when $2 \leqslant i \leqslant b-2$.

Remark 3. It should be noted that a_1 is still unknown using the strategy above. We can guess $a_1 = 0$ and $a_1 = 1$ respectively and then recover other bits.

To recover b_i, we need to query the verification oracle. Let

$$\boldsymbol{M}_i = \begin{pmatrix} 1 + 2^{b-i-2}\delta_i & 2^{b-i-1}\gamma_i \\ 0 & 1 - 2^{b-i-2}\delta_i \end{pmatrix} \text{ and } \boldsymbol{N}_i = \begin{pmatrix} 1 - 2^{b-i-2}\delta_i & 2^{b-i-1}\gamma_i \\ 0 & 1 + 2^{b-i-2}\delta_i \end{pmatrix}.$$

When $i \leqslant \frac{b}{2} - 1$, equivalently, $(2^{b-i-1})^2 = 0$ in $\mathbb{Z}/2^b\mathbb{Z}$, it can be computed that

$$\begin{aligned} \boldsymbol{N}_i \boldsymbol{A} \boldsymbol{M}_i^{-1} &= \begin{pmatrix} 1 - 2^{b-i-2}\delta_i & 2^{b-i-1}\gamma_i \\ 0 & 1 + 2^{b-i-2}\delta_i \end{pmatrix} \begin{pmatrix} \gamma & \delta \\ \delta & \gamma \end{pmatrix} \begin{pmatrix} 1 - 2^{b-i-2}\delta_i & -2^{b-i-1}\gamma_i \\ 0 & 1 + 2^{b-i-2}\delta_i \end{pmatrix} \\ &= \begin{pmatrix} \gamma + T_i & \delta \\ \delta & \gamma - T_i \end{pmatrix} \xlongequal{\text{Lemma 2}} \begin{pmatrix} \gamma + 2^{b-1} \cdot b_i & \delta \\ \delta & \gamma - 2^{b-1} \cdot b_i \end{pmatrix}, \end{aligned}$$

thus, $\boldsymbol{N}_i \boldsymbol{A} \boldsymbol{M}_i^{-1} = \boldsymbol{A}$ if and only if $b_i = 0$. We choose

$$\begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} = \boldsymbol{M}_i \begin{pmatrix} R_1 \\ S_1 \end{pmatrix}, \quad \begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} = \boldsymbol{N}_i \begin{pmatrix} R_2 \\ S_2 \end{pmatrix},$$

then $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1$ if and only if $b_i = 0$. It means that we can recover b_i through querying the verification oracle when $i \leqslant \frac{b}{2} - 1$. We summarize these steps as Algorithm 4 in Appendix A.

Case 2: $\boldsymbol{\gamma}$ is even and $\boldsymbol{\delta}$ is odd. In this case, γ_i is even and δ_i is odd for every $i \leqslant b - 1$.

We can recover b_i when $2 \leqslant i \leqslant b - 2$ similar to Case 1. b_1 is still unknown and we should assume $b_1 = 0$ and $b_1 = 1$ respectively.

To recover a_i, we also need to query the oracle. Let

$$\boldsymbol{M}_i = \begin{pmatrix} 1 + 2^{b-i-2}\gamma_i & 0 \\ 2^{b-i-1}\delta_i & 1 - 2^{b-i-2}\gamma_i \end{pmatrix} \text{ and } \boldsymbol{N}_i = \begin{pmatrix} 1 + 2^{b-i-2}\gamma_i & 2^{b-i-1}\delta_i \\ 0 & 1 - 2^{b-i-2}\gamma_i \end{pmatrix}.$$

When $i \leqslant \frac{b}{2} - 1$, equivalently, $(2^{b-i-1})^2 = 0$ in $\mathbb{Z}/2^b\mathbb{Z}$, it can be computed that

$$\boldsymbol{N}_i \boldsymbol{A} \boldsymbol{M}_i^{-1} = \begin{pmatrix} \gamma & \delta + T_i \\ \delta - T_i & \gamma \end{pmatrix} \xlongequal{\text{Lemma 2}} \begin{pmatrix} \gamma & \delta + 2^{b-1} \cdot a_i \\ \delta - 2^{b-1} \cdot a_i & \gamma \end{pmatrix},$$

thus, $\boldsymbol{N}_i \boldsymbol{A} \boldsymbol{M}_i^{-1} = \boldsymbol{A}$ if and only if $a_i = 0$. We choose

$$\begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} = \boldsymbol{M}_i \begin{pmatrix} R_1 \\ S_1 \end{pmatrix}, \quad \begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} = \boldsymbol{N}_i \begin{pmatrix} R_2 \\ S_2 \end{pmatrix},$$

then $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1$ if and only if $a_i = 0$. It means that we can recover a_i through querying the verification oracle when $i \leqslant \frac{b}{2} - 1$. We summarize these steps as Algorithm 5 in Appendix A.

Remark 4. This attack strategy is not applicable to a secret diagonal matrix. The determinant and verification oracle provide the same information in this case. Exploring an attack strategy for diagonal matrix is a topic for future research.

4.4 Analysis

Using the strategy above, we can recover the matrix $\boldsymbol{A}_1 \equiv \boldsymbol{A} \bmod 2^{\frac{b}{2}}$ through querying verification oracle. It should be noted that the determinants of all malicious matrices $\{M_i, N_i\}_{i=0}^{\frac{b}{2}-1}$ equal one in $\mathbb{Z}/2^b\mathbb{Z}$. Thus,

$$\begin{aligned} e_{2^b}(P_1, Q_1) &= e_{2^b}(R_1, S_1)^{\det \boldsymbol{M}_i} = e_{2^b}(R_1, S_1), \\ e_{2^b}(P_2, Q_2) &= e_{2^b}(R_2, S_2)^{\det \boldsymbol{N}_i} = e_{2^b}(R_2, S_2). \end{aligned}$$

It means that Weil pairing is unable to detect our choices of malicious torsion points. We conclude our adaptive attack in the following theorem.

Theorem 2. *Suppose the secret matrix in FESTA is circulant. Given FESTA's public parameter and public key $(p, E_0, P_b, Q_b, E_A, R_A, S_A)$, a honest ciphertext $(E_1,(R_1,\ S_1),E_2,(R_2,S_2))$, and a decryption machine, we can recover FESTA's secret key $(\boldsymbol{A}, \phi_A)$ in either $\frac{b}{2}$ or $b-2$ queries to the decryption machine using ciphertexts different from the challenge one. Moreover, FESTA PKE protocol with secret circulant matrix is not IND-CCA secure.*

Proof. From the analysis in Section 3, we can construct a verification oracle

$$O(E_1, (R_1, S_1), E_2, (R_2, S_2)) = \begin{cases} 1\ ,\ \text{if} \begin{pmatrix} R_2 \\ S_2 \end{pmatrix} = \boldsymbol{A} \cdot \dfrac{1}{d_1} \phi_2 \circ \phi_A \circ \widehat{\phi_1} \begin{pmatrix} R_1 \\ S_1 \end{pmatrix}, \\ 0\ ,\ \text{otherwise} \end{cases}$$

from the decryption machine via side channel. Querying the verification oracle once will call the decryption machine once.

From algorithm 3, 4, 5 and the analysis in Section 4.2 and 4.3, we know that matrix $\boldsymbol{A}_1 \equiv \boldsymbol{A} \bmod 2^{\frac{b}{2}}$ can be recovered through querying verification oracle. In this step we replace the honest cipertext with malicious torsion points, which means that we query the decryption machine with ciphertexts different from the challenge one. Write $\begin{pmatrix} P_A \\ Q_A \end{pmatrix} = \boldsymbol{A}_1^{-1} \begin{pmatrix} [2^{\frac{b}{2}}]R_A \\ [2^{\frac{b}{2}}]S_A \end{pmatrix}$, then the secret isogeny ϕ_A can be recovered using

$$\mathsf{TorAtk}(E_0, [2^{\frac{b}{2}}]P_b, [2^{\frac{b}{2}}]Q_b, E_A, P_A, Q_A, d_A).$$

Since $\begin{pmatrix} R_A \\ S_A \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} \phi_A(P_b) \\ \phi_A(Q_b) \end{pmatrix}$, the circulant matrix $\boldsymbol{A}$ can be recovered. Hence, we can recover FESTA's secret key $(\boldsymbol{A}, \phi_A)$.

Now, let's tally the number of queries to the decryption machine, which is equivalent to the number of queries to the verification oracle. Write matrix $\boldsymbol{A} = \begin{pmatrix} \gamma & \delta \\ \delta & \gamma \end{pmatrix}$, $\gamma = \sum\limits_{i=0}^{b-1} a_i 2^i$, $\delta = \sum\limits_{i=0}^{b-1} b_i 2^i$. The verification oracle is queried to recover $\boldsymbol{A}_1$, equivalently, a_i and b_i for $0 \leqslant i \leqslant \frac{b}{2} - 1$. To recover a_0 and b_0, we need one query to the oracle. Without loss of generality, suppose that γ is odd and δ is even, which is consistent with Case 1 in Section 4.3. To recover b_1, we need one query to the oracle. Then we guess $a_1 = 0$ and $a_1 = 1$ respectively to recover other bits. For $2 \leqslant i \leqslant \frac{b}{2} - 1$, we need one query to the oracle to recover (a_i, b_i). Thus, if the first guess of a_1 is right, then we only need $2 + (\frac{b}{2} - 2) = \frac{b}{2}$ queries in total. Otherwise, we need $2 + 2(\frac{b}{2} - 2) = b - 2$ queries in total. Note that $b < \log p$.

Therefore, given two messages and a cipertext encrypted by one of them, we can recover the secret key of FESTA in polynomial time. Then we can decrypt the cipertext directly to get corresponding message in polynomial time, because we have known the secret key. It follows that FESTA PKE protocol with secret circulant matrix is not IND-CCA secure. □

5 Implementation

We provide an implementation of our adaptive attack against FESTA PKE protocol in SageMath and make it available in the zip file:

attack-FESTA.zip

The TorAtk algorithm used in FESTA PKE protocol is limited to specific parameters, and there is no complete implementation of TorAtk algorithm. Therefore, our program focuses solely on recovering the matrix $\boldsymbol{A}_1 \equiv \boldsymbol{A} \bmod 2^{\frac{b}{2}}$ in our program, where $\boldsymbol{A}$ is the matrix in secret key. Note that in algorithm 4 and algorithm 5, we utilize the TorAtk algorithm to check the correctness of our second bit guesses. As an alternative, we introduce a comparison algorithm where we directly compare the recovered bits with the actual bits of the secret matrix.

Our implementation consists of two source files. All the following tests are conducted on a single performance core of an AMD Ryzen 7 7840H CPU.

- In the 'exception_time.py' file, we record the running time of two exceptions and define a function to compute the intermediate running time of two exceptions. The test results are presented in Table 1.

Table 1. Average running time of two exceptions

	Exception 1	Exception 2
Running time	6.768s	10.540s
	6.634s	10.463s
	6.658s	10.479s
	6.704s	10.545s
	6.648s	10.345
Average time	6.682s	10.474s

- In the 'attack_festa.sage' file, we recover the secret matrix in FESTA PKE protocol. There are two implementations of verification oracle presented in

Algorithm 1 and Algorithm 2, and we allow the choice of these two implementations as flag. Through a straightforward test outlined in Table 2, we know that average time to recover one bit is approximately 9 seconds. The number of bits to recover can be specified when running this file. If we choose $\frac{b}{2}$ bits, then we recover the matrix $\boldsymbol{A}_1 \equiv \boldsymbol{A} \bmod 2^{\frac{b}{2}}$. Against FESTA with 128-bit security, we successfully recover the secret matrix $\boldsymbol{A}_1$ in 2791.565s.

Table 2. Running time of attack algorithm

Number of bits recovered	Time
5	48.255s
10	87.869s
15	139.657s
20	183.680
25	230.195s

6 Conclusion

In this paper, We distinguished between two exceptions in the FESTA decryption algorithm through a side channel, constructing an effective verification oracle. By querying this oracle, we proposed a practical adaptive attack against FESTA PKE protocol, demonstrating its vulnerability to not being IND-CCA secure when the secret matrix is assumed to be circulant. Finally, we presented our attack algorithm and implemented it in SageMath.

We acknowledge the limitations of our attack strategy when dealing with secret diagonal matrices and propose future research directions, including the development of a constant-time implementation of the FESTA PKE protocol and the exploration of adaptive attacks targeting secret diagonal matrices.

A Algorithms

Algorithm 4: The other bits of secret matrix when γ is odd and δ is even

---

Input: The public parameter and public key $(p, E_0, P_b, Q_b, E_A, R_A, S_A)$, an honest ciphertext $(E_1, (R_1, S_1), E_2, (R_2, S_2))$, and the verification oracle O.

Output: $\gamma \bmod 2^{\frac{b}{2}}$, $\delta \bmod 2^{\frac{b}{2}}$.

Compute the determinant $\xi \leftarrow d_A^{-1} \cdot \mathbf{dlog}(e_{2^b}(R_A, S_A), e_{2^b}(P_b, Q_b))$;

Set $\gamma \leftarrow 1,\ \delta \leftarrow 0$;

Set $\binom{P_1}{Q_1} \leftarrow \begin{pmatrix} 1+2^{b-3}\delta & 2^{b-2}\gamma \\ 0 & 1-2^{b-3}\delta \end{pmatrix} \binom{R_1}{S_1}$, $\binom{P_2}{Q_2} \leftarrow \begin{pmatrix} 1-2^{b-3}\delta & 2^{b-2}\gamma \\ 0 & 1+2^{b-3}\delta \end{pmatrix} \binom{R_2}{S_2}$;

if $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1$ **then** set $b_1 \leftarrow 0$;

else set $b_1 \leftarrow 1,\ \delta \leftarrow \delta + b_1 \cdot 2$;

Set $a_1 \leftarrow 0$; ▷ guess $a_1 = 0$

for $i = 2 \rightarrow \frac{b}{2} - 1$ **do**

  if $\xi - \gamma^2 + \delta^2 \equiv 0 \pmod{2^{i+2}}$ **then** set $a_i \leftarrow 0$;

  else set $a_i \leftarrow 1$;

  Set $\binom{P_1}{Q_1} \leftarrow \begin{pmatrix} 1+2^{b-i-2}\delta & 2^{b-i-1}\gamma \\ 0 & 1-2^{b-i-2}\delta \end{pmatrix} \binom{R_1}{S_1}$, $\binom{P_2}{Q_2} \leftarrow \begin{pmatrix} 1-2^{b-i-2}\delta & 2^{b-i-1}\gamma \\ 0 & 1+2^{b-i-2}\delta \end{pmatrix} \binom{R_2}{S_2}$;

  if $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1$ **then** set $b_i \leftarrow 0$;

  else set $b_i \leftarrow 1$;

  Set $\gamma \leftarrow \gamma + a_i \cdot 2^i,\ \delta \leftarrow \delta + b_i \cdot 2^i$;

Set $\boldsymbol{A}_1 \leftarrow \begin{pmatrix} \gamma & \delta \\ \delta & \gamma \end{pmatrix}$, $\begin{pmatrix} P_A \\ Q_A \end{pmatrix} \leftarrow \boldsymbol{A}_1^{-1} \begin{pmatrix} [2^{\frac{b}{2}}]R_A \\ [2^{\frac{b}{2}}]S_A \end{pmatrix}$;

Set $\phi \leftarrow \mathsf{TorAtk}(E_0, [2^{\frac{b}{2}}]P_b, [2^{\frac{b}{2}}]Q_b, E_A, P_A, Q_A, d_A)$;

if $\phi \neq \perp$ **then return** γ, δ;

Set $a_1 \leftarrow 1, \gamma \leftarrow \gamma + a_1 \cdot 2$; ▷ guess $a_1 = 1$

for $i = 2 \rightarrow \frac{b}{2} - 1$ **do**

  if $\xi - \gamma^2 + \delta^2 \equiv 0 \pmod{2^{i+2}}$ **then** set $a_i \leftarrow 0$;

  else set $a_i \leftarrow 1$;

  Set $\binom{P_1}{Q_1} \leftarrow \begin{pmatrix} 1+2^{b-i-2}\delta & 2^{b-i-1}\gamma \\ 0 & 1-2^{b-i-2}\delta \end{pmatrix} \binom{R_1}{S_1}$, $\binom{P_2}{Q_2} \leftarrow \begin{pmatrix} 1-2^{b-i-2}\delta & 2^{b-i-1}\gamma \\ 0 & 1+2^{b-i-2}\delta \end{pmatrix} \binom{R_2}{S_2}$;

  if $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1$ **then** set $b_i \leftarrow 0$;

  else set $b_i \leftarrow 1$;

  Set $\gamma \leftarrow \gamma + a_i \cdot 2^i,\ \delta \leftarrow \delta + b_i \cdot 2^i$;

Set $\boldsymbol{A}_1 \leftarrow \begin{pmatrix} \gamma & \delta \\ \delta & \gamma \end{pmatrix}$, $\begin{pmatrix} P_A \\ Q_A \end{pmatrix} \leftarrow \boldsymbol{A}_1^{-1} \begin{pmatrix} [2^{\frac{b}{2}}]R_A \\ [2^{\frac{b}{2}}]S_A \end{pmatrix}$;

Set $\phi \leftarrow \mathsf{TorAtk}(E_0, [2^{\frac{b}{2}}]P_b, [2^{\frac{b}{2}}]Q_b, E_A, P_A, Q_A, d_A)$;

if $\phi \neq \perp$ **then return** γ, δ;

else return $\perp$;

Algorithm 5: The other bits of secret matrix when γ is even and δ is odd

Input: The public parameter and public key $(p, E_0, P_b, Q_b, E_A, R_A, S_A)$, an honest ciphertext $(E_1, (R_1, S_1), E_2, (R_2, S_2))$, and the verification oracle O.

Output: $\gamma \bmod 2^{\frac{b}{2}}$, $\delta \bmod 2^{\frac{b}{2}}$.

Compute the determinant $\xi \leftarrow d_A^{-1} \cdot \mathbf{dlog}(e_{2^b}(R_A, S_A), e_{2^b}(P_b, Q_b))$;

Set $\gamma \leftarrow 0,\ \delta \leftarrow 1$;

Set $\begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} \leftarrow \begin{pmatrix} 1 + 2^{b-3}\gamma & 0 \\ 2^{b-2}\delta & 1 - 2^{b-3}\gamma \end{pmatrix} \begin{pmatrix} R_1 \\ S_1 \end{pmatrix}$, $\begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} \leftarrow \begin{pmatrix} 1 + 2^{b-3}\gamma & 2^{b-2}\delta \\ 0 & 1 - 2^{b-3}\gamma \end{pmatrix} \begin{pmatrix} R_2 \\ S_2 \end{pmatrix}$;

if $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1$ **then** set $a_1 \leftarrow 0$;

else set $a_1 \leftarrow 1,\ \gamma \leftarrow \gamma + a_1 \cdot 2$;

Set $b_1 \leftarrow 0$; ▷ guess $b_1 = 0$

for $i = 2 \to \frac{b}{2} - 1$ **do**

- **if** $\xi - \gamma^2 + \delta^2 \equiv 0 \pmod{2^{i+2}}$ **then** set $b_i \leftarrow 0$;
- **else** set $b_i \leftarrow 1$;
- Set $\begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} \leftarrow \begin{pmatrix} 1 + 2^{b-i-2}\gamma & 0 \\ 2^{b-i-1}\delta & 1 - 2^{b-i-2}\gamma \end{pmatrix} \begin{pmatrix} R_1 \\ S_1 \end{pmatrix}$, $\begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} \leftarrow \begin{pmatrix} 1 + 2^{b-i-2}\gamma & 2^{b-i-1}\delta \\ 0 & 1 - 2^{b-i-2}\gamma \end{pmatrix} \begin{pmatrix} R_2 \\ S_2 \end{pmatrix}$;
- **if** $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1$ **then** set $a_i \leftarrow 0$;
- **else** set $a_i \leftarrow 1$;
- Set $\gamma \leftarrow \gamma + a_i \cdot 2^i,\ \delta \leftarrow \delta + b_i \cdot 2^i$

Set $\boldsymbol{A}_1 \leftarrow \begin{pmatrix} \gamma & \delta \\ \delta & \gamma \end{pmatrix}$, $\begin{pmatrix} P_A \\ Q_A \end{pmatrix} \leftarrow \boldsymbol{A}_1^{-1} \begin{pmatrix} [2^{\frac{b}{2}}]R_A \\ [2^{\frac{b}{2}}]S_A \end{pmatrix}$;

Set $\phi \leftarrow \mathsf{TorAtk}(E_0, [2^{\frac{b}{2}}]P_b, [2^{\frac{b}{2}}]Q_b, E_A, P_A, Q_A, d_A)$;

if $\phi \neq \perp$ **then return** γ, δ;

Set $b_1 \leftarrow 1, \delta \leftarrow \delta + b_1 \cdot 2$; ▷ guess $b_1 = 1$

for $i = 2 \to \frac{b}{2} - 1$ **do**

- **if** $\xi - \gamma^2 + \delta^2 \equiv 0 \pmod{2^{i+2}}$ **then** set $b_i \leftarrow 0$;
- **else** set $b_i \leftarrow 1$;
- Set $\begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} \leftarrow \begin{pmatrix} 1 + 2^{b-i-2}\gamma & 0 \\ 2^{b-i-1}\delta & 1 - 2^{b-i-2}\gamma \end{pmatrix} \begin{pmatrix} R_1 \\ S_1 \end{pmatrix}$, $\begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} \leftarrow \begin{pmatrix} 1 + 2^{b-i-2}\gamma & 2^{b-i-1}\delta \\ 0 & 1 - 2^{b-i-2}\gamma \end{pmatrix} \begin{pmatrix} R_2 \\ S_2 \end{pmatrix}$;
- **if** $O(E_1, (P_1, Q_1), E_2, (P_2, Q_2)) = 1$ **then** set $a_i \leftarrow 0$;
- **else** set $a_i \leftarrow 1$;
- Set $\gamma \leftarrow \gamma + a_i \cdot 2^i,\ \delta \leftarrow \delta + b_i \cdot 2^i$

Set $\boldsymbol{A}_1 \leftarrow \begin{pmatrix} \gamma & \delta \\ \delta & \gamma \end{pmatrix}$, $\begin{pmatrix} P_A \\ Q_A \end{pmatrix} \leftarrow \boldsymbol{A}_1^{-1} \begin{pmatrix} [2^{\frac{b}{2}}]R_A \\ [2^{\frac{b}{2}}]S_A \end{pmatrix}$;

Set $\phi \leftarrow \mathsf{TorAtk}(E_0, [2^{\frac{b}{2}}]P_b, [2^{\frac{b}{2}}]Q_b, E_A, P_A, Q_A, d_A)$;

if $\phi \neq \perp$ **then return** γ, δ;

else return $\perp$;

References

1. Albrecht, M.R., Bernstein, D.J., Chou, T., Cid, C., Gilcher, J., Lange, T., Maram, V., von Maurich, I., Misoczki, R., Niederhagen, R., Paterson, K.G., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., Szefer, J., Tjhai, C.J., Tomlinson, M., Wang, W.: Classic Mceliece (2022), https://classic.mceliece.org
2. Azarderakhsh, R., Campagna, M., Costello, C., De Feo, L., Hess, B., Hutchinson, A., Jalali, A., Jao, D., Karabina, K., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Pereira, G., Renes, J., Soukharev, V., Urbanik, D.: Supersingular isogeny key encapsulation (2020), http://sike.org
3. Basso, A., Maino, L., Pope, G.: Festa: Fast encryption from supersingular torsion attacks. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023. pp. 98–126. Springer Nature Singapore, Singapore (2023). https://doi.org/10.1007/978-981-99-8739-9_4
4. Castryck, W., Decru, T.: An efficient key recovery attack on sidh. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. pp. 423–447. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30589-4_15
5. Castryck, W., Vercauteren, F.: A polynomial time attack on instances of m-sidh and festa. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023. pp. 127–156. Springer Nature Singapore, Singapore (2023). https://doi.org/10.1007/978-981-99-8739-9_5
6. Costello, C., Hisil, H.: A simple and compact algorithm for SIDH with arbitrary degree isogenies. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 303–329. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_11
7. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: Sqisignhd: New dimensions in cryptography. Cryptology ePrint Archive, Paper 2023/436 (2023), https://eprint.iacr.org/2023/436
8. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology **8**(3), 209–247 (2014). https://doi.org/10.1515/jmc-2012-0015
9. Ebrahimi, E.: Post-quantum security of plain oaep transform. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public-Key Cryptography – PKC 2022. pp. 34–51. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-97121-2_2
10. Fouotsa, T.B., Petit, C.: SHealS and HealS: Isogeny-Based PKEs from a Key Validation Method for SIDH. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology ASIACRYPT 2021. vol. 13093, pp. 279–307. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-92068-5_10
11. Galbraith, S.D., Lai, Y.F.: Attack on sheals and heals: The second wave of gpst. In: Cheon, J.H., Johansson, T. (eds.) Post-Quantum Cryptography. pp. 399–421. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-17234-2_19
12. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. pp. 63–91. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_3
13. Galbraith, S.D., Petit, C., Silva, J.: Identification protocols and signature schemes based on supersingular isogeny problems. In: Takagi, T., Peyrin, T. (eds.) Advances

in Cryptology – ASIACRYPT 2017. pp. 3–33. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_1

14. Jao, D., De Feo, L.: Towards Quantum-Resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.Y. (ed.) Post-Quantum Cryptography. Lecture Notes in Computer Science, vol. 7071, pp. 19–34. Springer Berlin / Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_2
15. Kani, E.: The number of curves of genus two with elliptic differentials. Journal für die reine und angewandte Mathematik **485**, 93–122 (1997). https://doi.org/10.1515/crll.1997.485.93
16. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on sidh. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. pp. 448–471. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30589-4_16
17. Milne, J.S.: Abelian varieties. In: Cornell, G., Silverman, J.H. (eds.) Arithmetic Geometry. pp. 103–150. Springer New York, New York, NY (1986). https://doi.org/10.1007/978-1-4613-8655-1_5
18. Moriya, T.: The wrong use of festa trapdoor functions leads to an adaptive attack. Cryptology ePrint Archive, Paper 2023/1092 (2023), https://eprint.iacr.org/2023/1092
19. Mumford, D., Ramanujam, C.P., Manin, J.I.: Abelian varieties, vol. 5. Oxford university press Oxford (1974)
20. Petit, C.: Faster algorithms for isogeny problems using torsion point images. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 330–353. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_12
21. de Quehen, V., Kutas, P., Leonardi, C., Martindale, C., Panny, L., Petit, C., Stange, K.E.: Improved torsion-point attacks on SIDH variants. In: Advances in Cryptology – CRYPTO 2021. pp. 432–470. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-84252-9_15
22. Robert, D.: Breaking sidh in polynomial time. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. pp. 472–503. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30589-4_17
23. Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Seiler, G., Stehle, D., Ding, J.: Cryptographic suite for algebraic lattices (2019), https://pq-crystals.org
24. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review **41**(2), 303–332 (1999). https://doi.org/10.1137/S0036144598347011
25. Silverman, J.H.: The Arithmetic of Elliptic Curves, Graduate Texts in Mathematics, vol. 106. Springer New York, New York, NY (2009). https://doi.org/10.1007/978-0-387-09494-6
26. Tani, S.: Claw finding algorithms using quantum walk. Theoretical Computer Science **410**(50), 5285–5297 (2009). https://doi.org/10.1016/j.tcs.2009.08.030
27. Ti, Y.B.: Fault attack on supersingular isogeny cryptosystems. In: Lange, T., Takagi, T. (eds.) Post-Quantum Cryptography. pp. 107–122. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-59879-6_7