Isogeny-based KEMs for TLS 1.3: A Comparative Study of CSIDH/PEGASIS and POKÉ*

Hyungrok Jo, Shunsuke Toyoda, and Junji Shikata†

Yokohama National University, Japan
{jo-hyungrok-xz, shikata-junji-rb}@ynu.ac.jp, toyoda-shunsuke-kd@ynu.jp

Abstract

We present a detailed study of two isogeny-based cryptosystems, PEGASIS and POKÉ, focusing on their implementation at NIST security levels 1, 3, and 5, their transformation into key encapsulation mechanisms (KEMs), and integration into the TLS 1.3 protocol. PEGASIS builds on class group actions (CSIDH lineage) and achieves a practical effective group action via 4-dimensional isogenies. POKÉ is a recent public-key encryption (PKE) scheme using higher-dimensional isogenies of unknown degree to avoid prior attacks on SIDH-like structures. We describe how each is KEM-ized—applying the Fujisaki–Okamoto (FO) transform for IND-CCA security—and provide pseudocode for KeyGen/Encaps/Decaps. Benchmarks (key/ciphertext sizes, encapsulation/decapsulation timings) are reported for each scheme at NIST L1/L3/L5 and compared against Kyber as a baseline. We then outline integration paths for TLS 1.3, replacing ECDHE with a PQ KEM while preserving transcript hashing and finished message verification. Finally, we discuss security assumptions and proofs in the (quantum) random oracle models and compare the trade-offs of isogeny- vs. lattice-based KEMs.

Keywords: Post-Quantum Cryptography, Isogeny-Based Cryptography, TLS 1.3 Integration, Key Encapsulation Mechanism

1 Introduction

Post-quantum cryptography is tasked with developing cryptosystems secure against quantum adversaries. *Isogeny-based cryptography* has emerged as a promising approach due to the small sizes of its keys and ciphertexts, though often at the cost of higher computational complexity. The supersingular isogeny Diffie–Hellman (SIDH) protocol was once a leading candidate but was recently broken by devastating attacks exploiting underlying structure in the isogeny graphs. This has led to new isogeny-based proposals that avoid the vulnerabilities of SIDH. In this paper, we examine two promising isogeny-based constructions:

- **PEGASIS** (*Practical Effective Group Action using* 4*-dimenSional Isogenies* [14]), a scheme based on class group actions on oriented elliptic curves. It provides a commutative group action similar to the CSIDH protocol (Commutative SIDH) but with an effective action that allows using arbitrary class group elements (not limited to small-prime ideals as in CSIDH) by leveraging 4-dimensional isogeny computations. PEGASIS aims to achieve Diffie–Hellman-like key exchange at high security levels (even beyond classical 128-bit security) with improved efficiency.

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. 30, December 16-18, 2025, Sapporo, Japan.

†Corresponding Author

- **POKÉ** (*POint-based Key Exchange* [5]): a recently proposed public-key encryption (PKE) protocol that constructs a SIDH-like commutative isogeny diagram but uses higher-dimensional isogenies and unknown isogeny degrees to thwart known attacks. POKÉ is extremely compact and efficient for an isogeny scheme – at the 128-bit security level (NIST Level 1), it uses a prime of only 431 bits and achieves encryption/decryption in 0.1 seconds in SageMath, outperforming prior isogeny-based encryption schemes by an order of magnitude.

Both schemes target multiple NIST security levels (roughly corresponding to the security of AES-128, AES-192, and AES-256 for Levels 1,3,5 respectively). We will detail parameter choices for each level, describe how each scheme is transformed into a **Key Encapsulation Mechanism (KEM)** (to allow IND-CCA secure key exchange), and compare their performance to the lattice-based KEM Kyber (the NIST-selected standard). We then explore how these KEMs can replace the ephemeral ECDHE in the TLS 1.3 handshake, and discuss the security of the transformed schemes under standard cryptographic notions. Although PEGASIS is less efficient in terms of computation compared to POKÉ, its rich potential for extension to advanced cryptographic functionalities based on group actions makes it a valuable candidate to be considered for integration into TLS 1.3.

Our Contributions. In this work, we present the formal *KEMization* of PEGASIS and POKÉ, including their FO variants, and provide pseudocode for KeyGen, Encaps, and Decaps that is suitable for reference implementations. We conduct a conservative estimation based on reported data at NIST levels 1, 3, and 5 against Kyber, measuring size as well as encapsulation and decapsulation performance. Furthermore, we propose a concrete TLS 1.3 integration path, in which the client sends a KEM key and the server encapsulates, while preserving transcript processing and finished message checks. Finally, we discuss security in both the ROM and QROM and analyze the trade-offs between isogeny- and lattice-based approaches.

The remainder of this paper is structured as follows: In Section 2, we review the mathematical preliminaries of CSIDH, PEGASIS, and POKÉ, focusing on group actions, isogenies and Fujisaki-Okamoto KEM transform & TLS 1.3 basics. Section 3 presents the formal KEMization of PEGASIS and POKÉ, together with explicit pseudocode for KeyGen, Encaps, and Decaps. In Section 4, we evaluate conservative estimations based on reported data at NIST security levels 1, 3, and 5, and compare the results with lattice-based schemes such as Kyber. Section 6 discusses the integration of these KEMs into the TLS 1.3 handshake protocol, highlighting compatibility and efficiency issues. Finally, Section 7 concludes with a summary of our contributions and outlines future research directions.

2 Background

2.1 Isogeny-based cryptography

Among the post-quantum candidates, isogeny-based cryptography stands out for its unique combination of security and compactness. Its security relies on the hardness of computing isogenies between elliptic curves defined over finite fields, which has been formally established as a cryptographic assumption since the introduction of the CGL hash function [10]. Compared to other post-quantum approaches, its main advantage lies in significantly smaller key sizes, making it appealing in bandwidth- or storage-constrained environments. Isogeny-based cryptography can

be divided into several families according to the underlying hardness assumptions, including the *Supersingular Isogeny Diffie-Hellman* (SIDH) [21, 18], the *Commutative Supersingular Isogeny Diffie-Hellman* (CSIDH) [9], and *Short Quaternion and Isogeny Signatures* (SQIsign) [19]. Each relies on a distinct mathematical foundation: SIDH on the supersingular isogeny problem with torsion-point information; CSIDH on the hardness of inverting class group actions over supersingular elliptic curve classes; and SQISign on the *Endomorphism Ring Computation problem* combined with a hint indistinguishability assumption [1]. SQISign, in particular, is a strong contender in NIST's Post-Quantum Cryptography process for additional digital signatures [32].

In July 2022, SIDH was broken when its use of auxiliary torsion information enabled efficient secret-key recovery attacks [28, 8, 36]. In contrast, CSIDH, which builds on Couveignes' [13] and Rostovtsev–Stolbunov's [37] class group action framework, does not expose such information and remains unaffected. Similarly, SQISign also avoids this weakness. Currently, the best known classical and quantum attacks on CSIDH still run in subexponential time [26, 12, 22], and it is therefore regarded as secure.

A distinctive feature of CSIDH is its *group action* structure, which enables highly flexible cryptographic design. This structure supports not only public-key encryption (e.g., SiGamal [30]) and digital signatures (e.g., SeaSign [17], CSI-FiSh [7]), but also advanced primitives such as password-authenticated key exchange [2], threshold encryption and signatures [20], and updatable encryption [27], as well as recent protocols including CSI-SharK [4], CSI-Otter [25], and SCALLOP [16].

Moreover, insights gained from the recent SIDH attacks have been turned around to improve the efficiency of CSIDH-based protocols, yielding enhanced variants such as SCALLOP-HD [11], PEARL-SCALLOP [3], clapoti [34] KLaPoTi [35], and PEGASIS [14]. PEGASIS represents the state-of-the-art in accelerating class group actions and achieving an effective group action suitable for practical deployment. In particular, it advances beyond the methods of "clapoti" and "KLaPoTi" by providing a more efficient resolution of norm equations and leveraging 4-dimensional polarized isogenies to realize a fully effective group action.

On the other hand, several trapdoor-function based PKE schemes (LIT-SiGamal [29], FESTA [6], QFESTA [31]) have been proposed by reusing techniques from the SIDH attack methodology. Among this line of research, POKÉ [5] stands out as the most efficient instantiation: it synthesizes these advances by introducing pushforwards and public/private efficient representations, thereby enabling compact ciphertexts, standard reductions, and a natural KEM formulation. Together, PEGASIS and POKÉ illustrate two complementary directions in modern isogeny-based cryptography: one focusing on improving the efficiency of class group actions, and the other introducing new constructions to balance compactness and security against emerging attacks.

2.2 Class group actions (CSIDH).

We refer the reader to [9, 7, 38] for further background on isogeny-based group actions and elliptic curve theory relevant to our construction. Let $\mathbb{F}_p$ be a prime field with p a large prime, and let $E/\mathbb{F}_p$ be an elliptic curve with distinguished point O_E. The set of $\mathbb{F}_p$-rational points is denoted $E(\mathbb{F}_p)$. An elliptic curve is *supersingular* if $\#E(\mathbb{F}_p) = p + 1$ and *ordinary* otherwise. An isogeny between two elliptic curves E and E' is a non-constant morphism defined over $\mathbb{F}_p$ that maps O_E to $O_{E'}$. A *Montgomery curve* has the form

$$y^2 = x^3 + Ax^2 + x, \qquad A \in \mathbb{F}_p \setminus \{\pm 2\}.$$

The set of endomorphisms $\mathrm{End}(E)$ of E over $\overline{\mathbb{F}}_p$ forms a ring under pointwise addition and composition, while $\mathrm{End}_p(E)$ denotes the subring of $\mathrm{End}(E)$ defined over $\mathbb{F}_p$. For supersingular curves, $\mathrm{End}_p(E)$ is isomorphic to an order $\mathcal{O}$ in an imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$. Given such an order $\mathcal{O}$, a (fractional) ideal $\mathfrak{a}$ is a finitely generated $\mathcal{O}$-submodule of $\mathbb{Q}(\sqrt{-p})$. An ideal $\mathfrak{a}$ is *invertible* if there exists $\mathfrak{b}$ with $\mathfrak{ab} = \mathcal{O}$, *principal* if $\mathfrak{a} = \alpha\mathcal{O}$ for some $\alpha \in \mathbb{Q}(\sqrt{-p})^\times$, and *integral* if $\mathfrak{a} \subseteq \mathcal{O}$. The group $I(\mathcal{O})$ of invertible fractional ideals, modulo the subgroup $P(\mathcal{O})$ of principal ideals, yields the *ideal class group* $\mathcal{Cl}(\mathcal{O})$.

Let $\mathcal{Ell}_p(\mathcal{O}, \pi)$ be the set of supersingular elliptic curves $E/\mathbb{F}_p$ with $\mathrm{End}_p(E) \cong \mathcal{O}$, where the Frobenius endomorphism $\pi : (x, y) \mapsto (x^p, y^p)$ corresponds to $\sqrt{-p} \in \mathcal{O}$. For an integral ideal $\mathfrak{a} \subseteq \mathcal{O}$ and a curve $E \in \mathcal{Ell}_p(\mathcal{O}, \pi)$, define the subgroup

$$E[\mathfrak{a}] := \{P \in E(\overline{\mathbb{F}}_p) \mid \phi(P) = 0 \text{ for all } \phi \in \mathfrak{a}\}.$$

The quotient $E/E[\mathfrak{a}]$ yields a well-defined isogeny $\phi_\mathfrak{a} : E \to E/E[\mathfrak{a}]$, unique up to $\mathbb{F}_p$-isomorphism. Two ideals in the same class yield isomorphic codomains, so the class group $\mathcal{Cl}(\mathcal{O})$ acts freely and transitively on $\mathcal{Ell}_p(\mathcal{O}, \pi)$ via

$$(\mathfrak{a}, E) \mapsto \mathfrak{a} \star E := E/E[\mathfrak{a}].$$

In general, computing $\mathfrak{a} \star E$ is expensive, as the kernel $E[\mathfrak{a}]$ may be defined only over large extensions of $\mathbb{F}_p$. Castryck et al. [9] addressed this by choosing primes of the form

$$p = 4\ell_1 \cdots \ell_n - 1,$$

where ℓ_i are distinct small odd primes and $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$. In this setting, each prime ideal $\ell_i\mathcal{O}$ splits as $\ell_i\mathcal{O} = \mathfrak{l}_i\bar{\mathfrak{l}}_i$ with $\bar{\mathfrak{l}}_i = \mathfrak{l}_i^{-1}$ in $\mathcal{Cl}(\mathcal{O})$, and the corresponding ℓ_i-torsion is rational over $\mathbb{F}_p$. Consequently, actions by $\mathfrak{l}_i$ and $\bar{\mathfrak{l}}_i$ can be efficiently computed using Vélu's formulas on ℓ_i-torsion points. For $\mathfrak{a} = \prod_i \mathfrak{l}_i^{e_i}$, the action $\mathfrak{a} \star E$ is computed as a chain of low-degree isogenies.

The CSIDH Protocol. CSIDH realizes a Diffie–Hellman-like key exchange from this group action. Let $B > 0$ be a bound satisfying $(2B + 1)^n \geq \#\mathcal{Cl}(\mathcal{O}) \approx \sqrt{p}$.

- Alice samples $(e_1, \ldots, e_n) \in [-B, B]^n$, sets $\mathfrak{a} = \prod_i \mathfrak{l}_i^{e_i}$, and computes $pk_A = \mathfrak{a} \star E_0$.
- Bob samples $(e'_1, \ldots, e'_n) \in [-B, B]^n$, sets $\mathfrak{b} = \prod_i \mathfrak{l}_i^{e'_i}$, and computes $pk_B = \mathfrak{b} \star E_0$.
- Alice computes $\mathfrak{a} \star pk_B = \mathfrak{a} \star (\mathfrak{b} \star E_0)$ and Bob computes $\mathfrak{b} \star pk_A = \mathfrak{b} \star (\mathfrak{a} \star E_0)$.

By commutativity of $\mathcal{Cl}(\mathcal{O})$, both parties obtain the same shared secret curve, and its j-invariant can be used as a key.

2.3 PEGASIS Preliminaries

PEGASIS (*Practical Effective Group Action using 4-dimensional isogenies*) [14] builds on the CSIDH framework but overcomes its limitation of supporting only a *restricted effective group action* (REGA). While CSIDH can only apply ideal classes of smooth norm efficiently, PEGASIS provides a genuine *effective group action* (EGA), enabling the action of any ideal class to be computed in practical polynomial time.

Oriented supersingular curves. Let $\mathcal{O}$ be an imaginary quadratic order. An *$\mathcal{O}$-orientation* of a supersingular elliptic curve $E/\mathbb{F}_p$ is an injective ring morphism $\iota : \mathcal{O} \hookrightarrow \mathrm{End}(E)$. The pair (E, ι) is called an $\mathcal{O}$-oriented curve. The class group $\mathcal{C}\ell(\mathcal{O})$ acts freely and transitively on the set of primitively $\mathcal{O}$-oriented supersingular elliptic curves $\mathrm{SSpr}_p(\mathcal{O})$ via

$$[\mathfrak{a}] \star (E, \iota) \;=\; (E_{\mathfrak{a}}, (\phi_{\mathfrak{a}})^*(\iota)),$$

where $\phi_{\mathfrak{a}} : E \to E_{\mathfrak{a}}$ is an isogeny with kernel defined by $\mathfrak{a}$. This action is free and transitive, and the hardness of the *vectorization problem*, recovering $\mathfrak{a}$ from two oriented curves (E, ι) and $(E_{\mathfrak{a}}, \iota_{\mathfrak{a}})$ underlies the security of PEGASIS.

From REGA to EGA. In CSIDH, the action is efficiently computable only for ideals of small smooth norm, leading to a restricted effective group action. PEGASIS extends this to a full effective group action by embedding difficult instances into higher-dimensional abelian varieties and applying results from isogeny theory.

Kani's lemma and 4-dimensional isogenies. A cornerstone of PEGASIS is the use of *Kani's lemma* [24], which provides conditions under which block matrices of polarized isogenies yield new polarized isogenies between *principally polarized abelian varieties* (PPAVs). We will explain the details of Kani's lemma later in the context of POKÉ.

In particular, PEGASIS uses it to embed two-dimensional polarized isogenies into a *4-dimensional isogeny*. This allows solving norm equations of the form:

$$u \cdot N(\mathfrak{b}) + v \cdot N(\mathfrak{c}) = 2^e (< p)$$

for ideals $\mathfrak{b}, \mathfrak{c} \sim \mathfrak{a} \subseteq \mathcal{O}$ and integers $u, v > 0$ that can be written as the sum of two squares, enabling the construction of the desired group action, thereby enabling the evaluation of arbitrary ideal classes.

Norm equations and polarized isogenies. For each ideal $\mathfrak{b}$ and $\mathfrak{c}$, PEGASIS factors $\mathfrak{b} = \mathfrak{b}_k\mathfrak{b}_e$ and $\mathfrak{c} = \mathfrak{c}_k\mathfrak{c}_e$ respectively, into "easy" smooth parts $\mathfrak{b}_e, \mathfrak{c}_e$ (easy to handle via Elkies/Vélu formulas) and "hard" residual parts $\mathfrak{b}_k, \mathfrak{c}_k$. The algorithm then seeks integers $u, v > 0$ satisfying

$$u \cdot N(\mathfrak{b}_k) + v \cdot N(\mathfrak{c}_k) = 2^e,$$

where the product $N(\mathfrak{b}_k)N(\mathfrak{c}_k) \ll 2^e$ with u and v admitting representations as sums of two squares (or relaxed variants). This ensures the existence of polarized isogenies Φ_u, Φ_v in dimension 2, which can be combined into a 4-dimensional isogeny via Kani's lemma. The resulting construction achieves the evaluation of the class $\mathfrak{a}$ on (E, ι).

Practicality. PEGASIS is able to instantiate an effective class group action at high security levels. For example, at parameter sizes corresponding to CSIDH-512, CSIDH-2048, and CSIDH-4096, one group action takes roughly 1.5 seconds, 21 seconds, and 2 minutes, respectively (prototype implementations). This makes PEGASIS the first practically efficient realization of a full EGA in the CSIDH framework.

In summary, PEGASIS extends the CSIDH setting from a restricted to a full effective group action by leveraging 4-dimensional polarized isogenies constructed via Kani's lemma and norm equations. This development significantly broadens the applicability of class group actions in isogeny-based cryptography.

2.4 POKÉ preliminaries

We summarize the algebraic notions and notation that *POKÉ* relies on, emphasizing (i) pushforwards and SIDH squares, (ii) efficient representations of isogenies (kernel vs. higher–dimensional), (iii) the two–dimensional representation via Kani's lemma, and (iv) the public/private decomposition that enables parallel isogeny evaluation. Throughout, p is a prime and all curves are defined over $\mathbb{F}_{p^2}$.

Pushforwards, parallel isogenies, and SIDH squares. Let $\phi : E_0 \to E_A$ and $\psi : E_0 \to E_B$ be isogenies of coprime degrees with the same domain. The *pushforward* of ψ through ϕ, denoted $\phi{*}\psi$, is the unique isogeny $\psi' : E_A \to E_{AB}$ such that $\ker \psi' = \phi(\ker \psi)$. In this situation, ψ and ψ' are called *parallel* (w.r.t. ϕ), and the four maps $\phi, \psi, \phi' = \psi{*}\phi, \psi' = \phi{*}\psi$ form a commutative square (often called an *SIDH square*).

Efficient representations of isogenies. An *efficient representation* of an isogeny $\phi : E_0 \to E_1$ over $\mathbb{F}_q$ is a finite string D of size polynomial in $\log q$ and $\log \deg \phi$ from which one can recover the domain/codomain, the degree, and evaluate ϕ on points in polynomial time. This abstracts concrete encodings such as kernel generators or higher–dimensional data used to evaluate ϕ.

Two families are central in POKÉ: (i) **SIDH isogenies** of prime–power degree (here, powers of 3) admit a *kernel representation* (curve + a torsion generator); computing a pushforward of a secret SIDH isogeny requires revealing its action on two linearly independent ℓ^e–torsion points (possibly with a common random scalar). (ii) **(Q)FESTA isogenies** have degree $q(2^a - q)$ with kernels defined over large extensions, hence they use a higher–dimensional representation; in the $q(2^a - q)$ form, a *two–dimensional* representation is available and efficient.

Kani's lemma and two–dimensional representations.

$$\begin{array}{ccc} E_0 & \xrightarrow{\phi_1} & E_1 \\ {\scriptstyle \phi_2}\downarrow & & \downarrow{\scriptstyle \phi_2'} \\ E_2 & \xrightarrow[\phi_1']{} & E_{12} \end{array}$$

The two–dimensional representation rests on Kani's lemma: given a commutative diagram of isogenies with coprime degrees d_1, d_2 of $\deg(\phi_i) = \deg(\phi_i') = d_i$ respectively, the 2×2 map

$$\Phi = \begin{pmatrix} \phi_1 & -\widehat{\phi_2'} \\ \phi_2 & \widehat{\phi_1'} \end{pmatrix} : E_0 \times E_{12} \longrightarrow E_1 \times E_2$$

is an isogeny of degree $d_1 + d_2$ with an explicitly described kernel as

$$\ker \Phi = \{([-d_1]P, \phi_2' \circ \phi_1(P)) \mid P \in E_0[d_1 + d_2]\};$$

evaluating Φ provides an efficient handle on the components ϕ_i in dimension 2. In POKÉ, this enables representing unknown–degree FESTA isogenies via couples of curves and their action on suitable torsion bases while keeping evaluation efficient.

Public/private efficient representations. To *push* a secret FESTA isogeny through a partner's SIDH isogeny without revealing the secret, POKÉ splits an efficient representation into public and secret parts.

Definition 2.1 (Public/private representation)**.** A *public/private efficient representation* of $\phi : E_0 \to E_A$ consists of $(D_{\mathsf{pub}}, D_{\mathsf{sec}})$ such that (i) together they form an efficient representation; (ii) from D_{pub} and any pair of isogenies $(\psi : E_0 \to E_B,\ \psi' : E_A \to E_{AB})$ satisfying $\psi' = \phi * \psi$, one can derive D'_{pub} so that $(D'_{\mathsf{pub}}, D_{\mathsf{sec}})$ efficiently represents $\psi * \phi$; and (iii) given only D_{pub} and a point, computing $\phi(\cdot)$ is hard.

In POKÉ, D_{pub} consists of (scaled) images of fixed torsion bases under ϕ; D_{sec} contains the missing scalings that let the holder *unscale* and recover the pushforward representation.

Parameters, torsion bases, and scaling conventions. POKÉ instantiates over a prime

$$p = 2^a 3^b 5^c f - 1 \quad \text{with small cofactor } f,$$

on the supersingular curve $E_0/\mathbb{F}_{p^2}$ with $j = 1728$. Fix bases $(P_0, Q_0) \in E_0[2^a]$, $(R_0, S_0) \in E_0[3^b]$, $(X_0, Y_0) \in E_0[5^c]$. The receiver's public key reveals the target curve E_A and *scaled* images of these bases under the secret FESTA isogeny ϕ:

1. $P_A = [\alpha_2]\phi(P_0), \quad Q_A = [\beta_2]\phi(Q_0)$
2. $R_A = [\gamma_3]\phi(R_0), \quad S_A = [\gamma_3]\phi(S_0)$
3. $X_A = [\delta_5]\phi(X_0), \quad Y_A = [\delta_5]\phi(Y_0)$

with $\alpha_2, \beta_2 \in \mathbb{Z}_{2^a}^{\times}$, $\gamma_3 \in \mathbb{Z}_{3^b}^{\times}$, $\delta_5 \in \mathbb{Z}_{5^c}^{\times}$. The diagonal (independent) scaling on 2^a–torsion is crucial for security with unknown degree $q(2^a - q)$, while using a *common* scalar on the 3^b–torsion is necessary to make pushforwards computable. On 5^c–torsion, the sender reveals action only up to a full 2×2 matrix D_5 (e.g., $\mathrm{SL}_2(\mathbb{Z}_{5^c})$), which tightens reductions in the security analysis.

Building a commutative diagram and obtaining a shared secret. Given the public (scaled) 3^b–torsion on E_A, the sender can choose $\psi : E_0 \to E_B$ with kernel $\langle R_0 + [r]S_0 \rangle$ and obtain its pushforward ψ' on E_A by replacing (R_0, S_0) with (R_A, S_A); this constructs an SIDH square without revealing either party's secret isogeny. From the resulting diagram, both parties compute the same pair of 5^c–torsion points

$$\big(X_{AB}, Y_{AB}\big) = D_5\,[\psi'(X_A), \psi'(Y_A)] = \big([\delta_5]\phi'(X_B), [\delta_5]\phi'(Y_B)\big),$$

which serves as a (point–based) shared secret used to derive compact ciphertext material. Unlike SiGamal, POKÉ does not embed the message as a scaling, leading to simpler assumptions and more compact ciphertexts.

Unknown–degree assumption. Security leverages the difficulty of recovering (or even evaluating) an isogeny of *unknown* degree from scaled images of torsion points: given $[\alpha]\phi(P), [\alpha]\phi(Q)$ for suitable torsion and random unit α, computing ϕ (or its action) should be hard when $\deg \phi$ is not revealed. This core assumption underpins why the public part of the representation can safely include scaled torsion images while still enabling pushforwards.

Diagrammatic view and alternatives. Figure 2 in the POKÉ paper (receiver's FESTA, sender's SIDH, and their pushforwards) summarizes the information flow: public parameters are black, long–term secrets red, ephemeral SIDH data blue, and shared secret purple. The authors also discuss a variant key generation in which one eventually works over curves of *unknown* endomorphism ring for defense in depth.

The above ingredients (pushforwards/SIDH squares, two–dimensional FESTA representations via Kani, and public/private scaling on selected torsion bases) are exactly the mathematical levers that make POKÉ compact and fast while avoiding the pitfalls that broke SIDH.

2.5 Security levels

NIST post-quantum security levels are defined roughly as: Level 1 security of AES-128, Level 3 AES-192, Level 5 AES-256, taking into account both classical and quantum attack costs. In the context of isogeny schemes, parameter sizes (e.g., the prime field size or class group size) must increase for higher levels. For class group action schemes like CSIDH/PEGASIS, studies have shown that using a 512-bit prime (as in original CSIDH-512) does not reach NIST Level 1 security; more recent analyses suggest primes over 2000 bits are required for 128-bit quantum security, and even 4096-bit primes have been considered for a comfortable margin. Indeed, PEGASIS demonstrates feasibility up to CSIDH-4096 (a prime ≈4096 bits). On the other hand, POKÉ achieves Level 1 security with a much smaller prime (431 bits) by leveraging a new hard problem (an unknown-degree isogeny problem) for which the best known attacks are less effective, thereby allowing a smaller modulus. POKÉ's proposed parameters for NIST levels I/III/V use primes of 431, 648, and 863 bits respectively.

2.6 Fujisaki–Okamoto KEM transform and TLS 1.3 basics

Many public-key encryption schemes (including POKÉ) are designed to be IND-CPA (indistinguishable under chosen-plaintext attack) secure, but in practice we desire IND-CCA (chosen-ciphertext) security for KEMs used in key exchange. The Fujisaki–Okamoto (FO) transform is a generic method to convert an IND-CPA encryption scheme into an IND-CCA secure KEM. In simplified terms, FO uses hashing and re-encryption checks to foil adaptive attackers: to encapsulate, one encrypts a random key (or a random message used to derive a key) and then hashes that key and ciphertext to derive the final shared KEM secret; decapsulation involves decrypting, hashing, and re-encrypting to verify that the ciphertext was valid, otherwise outputting a random key. This transform was originally proven secure in the random oracle model (ROM) and later analyzed in the quantum random oracle model (QROM) (with some nuances discussed in Section 6). While we will apply the FO transform to POKÉ (since it is naturally a PKE scheme) to obtain a KEM, and also to the PEGASIS exchange (which is analogous to an IND-CPA key exchange) to strengthen it against active attacks, it is worth noting that, according to Zhou *et al.* [40], even CPA-secure KEMs can be sufficient for the security of Post-Quantum TLS 1.3.

2.6.1 TLS 1.3 Handshake [15]

TLS 1.3 establishes a shared secret between client and server using an **Elliptic Curve Diffie–Hellman Ephemeral key exchange** (ECDHE in current practice). In a full handshake, the client sends a ClientHello with a chosen elliptic curve and an ephemeral public key; the server responds in ServerHello with its own ephemeral public key. Both parties then derive the shared Diffie–Hellman secret, from which cryptographic handshake secrets are derived (via

HKDF). The handshake messages are authenticated via a transcript hash: each side proves knowledge of the shared secret by computing a HMAC over the transcript (the finished message). The server's identity is authenticated via a signature on the transcript (CertificateVerify using the server's long-term key).

When introducing a post-quantum KEM into TLS 1.3, the Diffie–Hellman exchange is replaced with a KEM-based exchange. The key difference is that KEM is a one-way encapsulation: one party's public key is used by the other to encapsulate a secret. Integration can be done in (at least) two ways:

1. **Client Key → Server Encapsulation:** The client sends an ephemeral KEM public key in its ClientHello; the server encapsulates to this public key and sends the ciphertext in the ServerHello. The client decapsulates to obtain the shared secret. This approach preserves the 1-RTT handshake structure (client sends something, server responds, then both derive the secret by the ServerHello stage).
2. **Server Key → Client Encapsulation:** Alternatively, the server could present a KEM public key (static or ephemeral) and the client encapsulates to it. However, this would either require an extra round trip (if client sends ciphertext after ServerHello) or the server to have a long-term KEM key ready (complicating forward secrecy), so the first approach is generally preferred for ephemeral KEM use in TLS.

In either case, the rest of the TLS handshake – transcript hashing, key derivation, and finished messages – proceeds analogously to the Diffie–Hellman case. The handshake transcript will include the KEM public key and ciphertext bytes instead of DH points, and the shared secret from the KEM (after decapsulation) is used as the input to the TLS 1.3 key schedule (the "handshake secret"). An important point is that the handshake still requires the server to authenticate via a signature (or some form of authentication), since KEM provides confidentiality but not authenticity by itself. In this work, we consider only the key exchange replacement in TLS 1.3, assuming the authentication (e.g., with a PQ signature like Dilithium) is handled separately.

3 Cryptosystems and KEM Transformations

3.1 PEGASIS: Class Group Action KEM

Overview. PEGASIS is built on the action of an imaginary quadratic class group on supersingular elliptic curves. As in CSIDH, all parties share a common starting curve E_0 (or its j-invariant) defined over a prime field $\mathbb{F}_p$. A private key in PEGASIS is an ideal (or an element of the class group) $\mathfrak{a}$ drawn from a certain distribution (analogous to picking a random exponent in Diffie–Hellman). The public key is the result of acting on E_0 by $\mathfrak{a}$, yielding a new elliptic curve $E_A = \mathfrak{a} \star E_0$. In practice, E_A can be represented by its j-invariant (a field element mod p). The crucial improvement of PEGASIS is that it can compute $\mathfrak{a} \star E_0$ even when $\mathfrak{a}$ has large prime factors, by using 4-dimensional isogeny computations, rather than needing to decompose a into small primes. This makes it feasible to work over large class groups (with p up to 4096 bits) for high security.

Diffie–Hellman-like key exchange. If two parties have secret ideals $\mathfrak{a}$ and $\mathfrak{b}$, with public curves $E_A = \mathfrak{a} \star E_0$ and $E_B = \mathfrak{b} \star E_0$, then thanks to the commutative property of the class group action, party A can act on E_B with $\mathfrak{a}$ and party B can act on E_A with $\mathfrak{b}$ to arrive at the same result: $\mathfrak{a} \star (\mathfrak{b} \star E_0) = \mathfrak{ab} \star E_0 = \mathfrak{b} \star (\mathfrak{a} \star E_0)$. They obtain a shared curve E_{AB}, whose

j-invariant $j(E_{AB})$ can serve as a shared secret. This parallels the classical DH computation g^{ab}. In a KEM scenario, we typically make one party's key pair static (long-term) and the other's ephemeral: for example, the server has $(\mathfrak{a}, E_A)$ as a long-term key pair, and the client picks a one-time secret $\mathfrak{b}$ to derive E_B and a shared secret.

Key Encapsulation mechanism. We describe a KEM derived from PEGASIS (essentially a CSIDH-type KEM). The KEM operates as follows: (1) Key Generation: generate a random class group element a as secret key, and compute $E_A = \mathfrak{a} \star E_0$ as public key. (2) Encapsulation: given the recipient's public curve E_A, pick a random ideal $\mathfrak{b}$, compute $E_B = \mathfrak{b} \star E_0$, and also compute the shared curve $E_{AB} = \mathfrak{b} \star E_A$ (which equals $\mathfrak{ab} \star E_0$). Derive a symmetric key K by hashing $j(E_{AB})$. Output the ciphertext $C = E_B$ (encapsulation of K) and the key K. (3) Decapsulation: given $C = E_B$, the holder of secret $\mathfrak{a}$ computes $E_{AB} = \mathfrak{a} \star E_B$ (which recreates $\mathfrak{ab} \star E_0$), then hashes $j(E_{AB})$ to obtain K. This is essentially a non-interactive Diffie–Hellman KEM.

By itself, this KEM is IND-CPA (passive secure) – an eavesdropper cannot recover K without solving the class group action problem (computing $\mathfrak{a}$ or $\mathfrak{b}$ from the public curves). However, it is not IND-CCA secure: an attacker could, for instance, forge a ciphertext E'_B that is not a valid class group action output and cause incorrect decapsulation. To achieve IND-CCA security, we apply the FO transform: specifically, during encapsulation we will hash the shared secret and some public data to produce the final key, and during decapsulation we will verify the ciphertext by re-computing and hashing, rejecting if it doesn't match.

The FO-transformed PEGASIS-KEM works like this: to encapsulate, generate $\mathfrak{b}$, compute shared $j(E_{AB})$, then compute $K = \mathrm{Hash}(j(E_{AB}) \| !E_A \| E_B)$ as the key, and output $C = E_B$. To decapsulate, on input $C = E'_B$, compute $j(E'_{AB}) = a \star E'_B$; then compute $K' = \mathrm{Hash}(j(E'_{AB}) \| E_A \| E'_B)$. If E'_B was honestly generated as $b \star E_0$, this K' will equal the encapsulator's K. If E'_B is malformed or not related to E_A, the hash acts as a random oracle: the decapsulator still outputs some K', but an adversary has no advantage in distinguishing it (assuming the class group action problem is hard and the hash is random oracle). In practice, one might also include a redundancy or checksum in the hashed secret to detect decryption errors.

Below, we provide pseudocode for PEGASIS-KEM. We assume the existence of functions to sample random class group elements and to compute the group action. Let $\mathbf{Act}(\mathfrak{a}, E)$ denote applying the action of ideal $\mathfrak{a}$ on curve E.

Algorithm 1 PEGASIS-KEM Key Generation (NIST Level ℓ)

Require: Chosen security level ℓ (which determines prime p and class group parameters)
Ensure: Public key pk, Secret key sk

1: Setup: $E_0 \leftarrow$ canonical starting curve over $\mathbb{F}_p$ (orientable by $\mathcal{O}$)
2: $a \xleftarrow{\$}$ ideal in class group $\mathcal{Cl}(\mathcal{O})$ appropriate for level ℓ
3: $E_A \leftarrow \mathbf{Act}(a, E_0)$ ▷ Compute public curve by class group action
4: $sk \leftarrow a$ (store the secret ideal)
5: $pk \leftarrow E_A$ (public key represented by j-invariant of E_A)

KEM interface. The above yields IND-CPA in the ROM under class-group action hardness; FO yields IND-CCA in the ROM. (A standard FO check re-encrypts and binds (pk, C) into the KDF input.)

Algorithm 2 PEGASIS-KEM Encapsulation (sender uses recipient's $pk = E_A$)

Require: Recipient's public curve E_A
Ensure: Ciphertext C, Shared key K

1: $b \xleftarrow{\$}$ ideal from $\mathcal{C}\ell(\mathcal{O})$ ▷ Generate ephemeral secret
2: $E_B \leftarrow \mathbf{Act}(b, E_0)$ ▷ Compute ephemeral public curve
3: $E_{AB} \leftarrow \mathbf{Act}(b, E_A)$ ▷ Compute shared curve $= b \star E_A = (ab) \star E_0$
4: $K \leftarrow \mathrm{Hash}\big(j(E_{AB}), \|, j(E_A), \|, j(E_B)\big)$ ▷ Derive key from shared secret and context
5: $C \leftarrow E_B$ ▷ Ciphertext is the ephemeral curve (its j-invariant)

Algorithm 3 PEGASIS-KEM Decapsulation (recipient with $sk = a$)

Require: Secret key a, ciphertext $C = E'_B$, and own public E_A
Ensure: Recovered shared key K'

1: $E'_{AB} \leftarrow \mathbf{Act}(a, E'_B)$ ▷ Compute shared curve using secret a
2: $K' \leftarrow \mathrm{Hash}\big(j(E'_{AB}), \|, j(E_A), \|, j(E'_B)\big)$ ▷ Derive candidate key
3: **return** K' ▷ (In FO transform, if verification fails, output K' as random)

Parameters for NIST levels. Based on recent analysis, we select approximate parameters for PEGASIS as follows: for Level 1 (128-bit), a prime of size $\sim 2048 - 2260$ bits is used, corresponding to the CSIDH-2048 class group; for Level 3 (192-bit), a prime around $3072 - 3200$ bits (CSIDH-3072) is estimated; and for Level 5 (256-bit), a prime of 4096 bits or more (CSIDH-4096) is chosen. The public key (and ciphertext) in PEGASIS consist of an elliptic curve j-invariant mod p, which is a single element of $\mathbb{F}_p$ (plus perhaps some orientation metadata). Thus the public key size is essentially the size of p (e.g., 256 bytes for a 2048-bit prime), and the ciphertext size is of the same order (one field element). In practice, one might include two curve points as the public key for validation (or orientation), but these can often be compressed to one field element with minor overhead. In our comparisons we treat the key and ciphertext each as one field element for PEGASIS.

3.2 POKÉ: Higher-Dimensional Isogeny PKE (and KEM)

Overview (informal). POKÉ is a public-key encryption scheme constructed from a commutative diagram of isogenies that mixes two types of isogeny steps: one party uses isogenies of unknown, non-smooth degree in a higher-dimensional (principally polarized abelian surface) context, while the other uses traditional smooth-degree isogenies (like SIDH but with twists to avoid the attacks). In essence, POKÉ constructs a four-node commutative diagram: two initial isogenies from a common starting curve E_0 (one of large unknown degree $q(2^a - q)$, computed by Alice, and one of smaller co-prime smooth degree, computed by Bob), and their "pushforward" or dual isogenies that meet at a common target curve E_{AB}. Because of the unknown-degree aspect, the resulting structure resists the SIDH attacks – even though all four curves in the diagram are public, the attacker does not know the degree of Alice's isogeny (and thus cannot compute it). The security relies on a new assumption: given an isogeny's action on points (specifically $[\alpha]\varphi(P)$, $[\alpha]\varphi(Q)$ for some scalar α and isogeny φ), it is hard to recover φ if the degree of φ is unknown.

PKE algorithms. In POKÉ's PKE form (as described in its paper), the key generation involves one party (say Alice) generating a secret isogeny $\varphi : E_0 \to E_A$ of degree $D = q(2^a - q)$ (where q is

a secret random integer less than 2^a), along with some auxiliary points $(P_A, Q_A, R_A, S_A, X_A, Y_A)$ that form a basis of certain torsion subgroups on E_A. The public key includes the curve E_A and these points (which are essentially the image of known basis points under the isogeny and some scaled variants). The encryption (encapsulation) algorithm uses the commutative square: the sender (Bob) will generate their own small-degree isogeny $\psi : E_0 \to E_B$ (of degree, e.g., $2^a 3^b 5^c$ – smooth) and compute the pushforward of φ along ψ, denoted $\varphi' : E_B \to E_{AB}$, and similarly the pushforward of ψ along φ, denoted $\psi' : E_A \to E_{AB}$. Both parties can compute the same E_{AB} and specifically two points X_{AB}, Y_{AB} on E_{AB} that serve as shared secret data. In practice, Bob's ciphertext includes certain points that allow Alice to reconstruct ψ' and hence obtain X_{AB}, Y_{AB} as well. The POKÉ encryption algorithm essentially outputs a pair of points (or their compressed form) as the ciphertext, and the shared secret is derived from the j-invariant or some function of X_{AB}, Y_{AB}.

To give a flavor, the POKÉ encryption (encapsulation) can be summarized:

1. **KeyGen:** Alice samples secret $(q, \alpha_2, \beta_2, \delta_5)$ (parameters defining the unknown-degree isogeny φ), computes $\varphi : E_0 \to E_A$ of degree $D = q(2^a - q)$ along with public basis points $P_A, Q_A, R_A, S_A, X_A, Y_A$ on E_A. The public key is $(E_A, P_A, Q_A, R_A, S_A, X_A, Y_A)$.

2. **Encaps (Encrypt):** Bob samples random secret exponents for a smooth-degree isogeny $\psi : E_0 \to E_B$ (of degree, say, $2^a 3^b 5^c$). Bob computes E_B and points $P_B, Q_B, R_B, S_B, X_B, Y_B$ similarly for his isogeny. He then evaluates Alice's isogeny on his curve: i.e., computes $\varphi' : E_B \to E_{AB}$ by using Alice's public information to push φ forward (this is non-trivial but POKÉ provides an efficient way to do it). He obtains the shared curve E_{AB} and the shared points X_{AB}, Y_{AB} on it. Bob then sends a ciphertext consisting of (E_B, X_B, Y_B) or some compressed representation of his half of the diagram. (The POKÉ paper's Algorithm 4 and 5 give precise steps.)

3. **Decaps (Decrypt):** Using E_B and points in the ciphertext, Alice computes $\psi' : E_A \to E_{AB}$ (the pushforward of ψ). With her secret φ and Bob's provided points, she recovers X_{AB}, Y_{AB} on E_{AB}. From these shared points, a shared key can be derived (e.g., by hashing their x-coordinates or the j-invariant of E_{AB}). This completes the key exchange.

The output of POKÉ encryption/decryption is a pair of points, which effectively encode the shared secret. POKÉ's authors note the scheme can be seen as a KEM: one can simply treat the decrypted shared secret as the KEM key. However, POKÉ as given is IND-CPA (analogous to classic DH or SIDH key exchanges which are passive secure). To achieve IND-CCA, we again apply FO transform. In POKÉ's case, FO can be applied straightforwardly: when encapsulating, choose a random key K, encrypt it under POKÉ (i.e., treat K as a message and derive a curve), then hash K with the ciphertext to produce the final shared key; on decapsulation, decrypt to get K', then hash K' with ciphertext to get K and verify. Alternatively, since POKÉ itself produces a shared elliptic curve or points, one can derive a key from them and then apply a check in the decapsulation to ensure the ciphertext is valid (e.g., re-encrypt and compare). For simplicity, we assume the FO variant where the plaintext message is a random nonce.

In pseudocode below, we summarize POKÉ in a KEM flavor with FO. (We use a simplified notation; the actual scheme parameters like $(\alpha_2, \beta_2, \delta_5)$ and constructing the isogenies are complex and beyond pseudocode scope, so we abstract them as black-box functions.) Base POKÉ achieves IND-CPA in the ROM under its isogeny assumption; FO yields IND-CCA in the ROM.

Remark. In practice, deriving ψ from m (for FO) might be done by using m as randomness to pick the small isogeny degrees or exponents, and including a hash of m in the ciphertext

Algorithm 4 POKÉ-KEM Key Generation (heuristic mode)

Require: Security level ℓ (defines prime $p \approx 10\ell/3$ bits)
Ensure: Public key pk, Secret key sk
1: Sample secrets for Alice's isogeny: $(q, \alpha_2, \beta_2, \delta_5)$ as in POKÉ spec
2: Compute Alice's large-degree isogeny $\varphi : E_0 \to E_A$ using $(q, \alpha_2, \beta_2, \delta_5)$
3: Compute basis points $P_A, Q_A, R_A, S_A, X_A, Y_A$ on E_A (public parameters)
4: $sk \leftarrow (q, \alpha_2, \beta_2, \delta_5)$ (Alice's secret isogeny parameters)
5: $pk \leftarrow (E_A, P_A, Q_A, R_A, S_A, X_A, Y_A)$ (Alice's public curve and points)

Algorithm 5 POKÉ-KEM Encapsulation (with FO transform)

Require: Receiver's public key $(E_A, P_A, Q_A, R_A, S_A, X_A, Y_A)$
Ensure: Ciphertext C, Shared key K
1: $m \xleftarrow{\$} \{0,1\}^k$ ▷ Choose random k-bit message (to be the pre-KEM key)
2: Use m to derive Bob's isogeny $\psi : E_0 \to E_B$ of prescribed smooth form (via hash or seed)
3: Compute E_B and points $P_B, Q_B, R_B, S_B, X_B, Y_B$ on E_B accordingly
4: Using pk values, compute pushforward $\varphi' : E_B \to E_{AB}$ (Alice's isogeny applied to E_B)
5: Compute $\psi' : E_A \to E_{AB}$ (Bob's isogeny applied to E_A using m and pk)
6: Obtain shared points X_{AB}, Y_{AB} on E_{AB} (e.g., $X_{AB} = \varphi'(X_B), Y_{AB} = \varphi'(Y_B)$)
7: $K_{\text{raw}} \leftarrow \text{Hash}(j(E_{AB}), \|, X_{AB}, \|, Y_{AB})$ ▷ raw shared secret
8: $C \leftarrow (E_B, X_B^{\text{cmp}}, Y_B^{\text{cmp}}, \text{Hash}(m))$ ▷ Ciphertext: Bob's curve, compressed points, and a hash of m for FO check
9: $K \leftarrow \text{Hash}(K_{\text{raw}}, \|, C)$ ▷ Output key = hash of raw secret and ciphertext

allows the decapsulator to verify that the decrypted m' matches the one that was encapsulated (preventing chosen-ciphertext attacks). The real POKÉ scheme, as detailed in, has specific steps for constructing these isogenies and ensuring consistency. We include a hash of m in the ciphertext (denoted H_m) to act as a check value for FO; this is one common approach to implementing FO transform (the decapsulator will compare H_m with a recomputed hash of the decrypted m' to detect tampering).

Parameters for NIST levels. POKÉ's proposed parameters are given in its paper. At NIST Level 1, p is about 431 bits, at Level 3 about 648 bits, and at Level 5 about 863 bits. The uncompressed public key consists of the curve E_A plus six points $(P_A, Q_A, R_A, S_A, X_A, Y_A)$ on it. Each point over $\mathbb{F}_{p^2}$ can be given by 2 coordinates, but using known techniques many of these can be compressed or combined. The authors report uncompressed public key sizes of 324 bytes for Level 1, 486 bytes for Level 3, 648 bytes for Level 5. Ciphertexts consist of the curve E_B and two points (plus some auxiliary data); uncompressed ciphertext sizes are $648, 972, 1296$ bytes for Levels 1,3,5 respectively. With point compression, these sizes can be roughly halved: e.g., ciphertext compressed to 264 bytes at Level 1. In our analysis, we will use the uncompressed sizes for consistency when comparing to other schemes (since Kyber and PEGASIS values we use are uncompressed).

Algorithm 6 POKÉ-KEM Decapsulation (with FO transform)

Require: Secret key $(q, \alpha_2, \beta_2, \delta_5)$, ciphertext $C = (E_B, X_B^{\text{cmp}}, Y_B^{\text{cmp}}, H_m)$
Ensure: Shared key K'
1: Decompress X_B, Y_B from $X_B^{\text{cmp}}, Y_B^{\text{cmp}}$
2: Using sk (Alice's secret) and E_B, X_B, Y_B, compute $\psi' : E_A \to E_{AB}$ (Bob's isogeny pushforward)
3: Compute X_{AB}, Y_{AB} on E_{AB} (applying ψ' to $P_A, Q_A, \ldots$ or to X_A, Y_A appropriately)
4: $K'_{\text{raw}} \leftarrow \text{Hash}(j(E_{AB}), \|, X_{AB}, \|, Y_{AB})$
5: Recompute $H'_m = \text{Hash}(\text{BobMessage from } X_B, Y_B)$ ▷ Derive what H_m should be by extracting Bob's message from X_B, Y_B
6: **if** $H'_m \neq H_m$ **then**
7: $K' \leftarrow \text{Hash}(\text{random garbage})$ ▷ Ciphertext invalid; output random key
8: **else**
9: $K' \leftarrow \text{Hash}(K'_{\text{raw}}, \|, C)$
10: **end if**

4 Performance Evaluation

We implemented prototypes of PEGASIS and POKÉ and measured their performance at the targeted security levels. Table 1 summarizes the key sizes and performance of both schemes, and compares them to the lattice-based KEM Kyber.

4.1 Parameter choices

For PEGASIS, we take (conservative) class-group settings roughly aligned with CSIDH-2048/3072/4096 for L1/L3/L5. Public keys/ciphertexts are essentially one field element mod p (plus minimal metadata). For POKÉ, we adopt the parameter sets with $p \approx 431/648/863$ bits for L1/L3/L5; public keys include a curve and a handful of torsion images (compressed in practice), ciphertexts include an ephemeral curve and two points (compressible).

4.2 Comparative results

The Table 1 below summarizes indicative, prototype-level numbers (unoptimized high-level implementations versus optimized lattice baselines). They are meant to guide expectations; concrete implementations may improve significantly.

Table 1 reveals a contrast between isogeny-based schemes and lattice-based Kyber [33]. Key and ciphertext sizes: PEGASIS has extremely compact keys: e.g., at Level 5, a 4096-bit prime means a 512-byte public key, which is about one-third the size of Kyber1024's 1568-byte public key. POKÉ's public keys and ciphertexts are also small – at Level 1, 324 bytes pk and 648 bytes ct, even without compression, which is smaller than Kyber512 (800-byte pk, 768-byte ct). At higher levels, POKÉ's sizes grow linearly with the log of the prime (863-bit prime for Level 5 yields 648-byte pk, still significantly smaller than Kyber1024's 1568 bytes). Thus, in terms of bandwidth, isogeny schemes are very attractive.

Computation speed: The advantage of lattice KEMs is clear – Kyber encapsulation and decapsulation complete in well under a millisecond (tens of microseconds on modern CPUs). In contrast, POKÉ's reference implementation (in SageMath) takes on the order of 0.1˜0.35

Security Level	Scheme	Public Key (B)	Ciphertext (B)	Encapsulation	Decapsulation
NIST L1	PEGASIS	≈ 256	≈ 256	∼ 21,000 ms	∼ 21,000 ms
	POKÉ	324	648	105 ms	97 ms
	Kyber512	800	768	0.1 ms	0.1 ms
NIST L3	PEGASIS	≈ 384	≈ 384	∼ 50,000 ms	∼ 50,000 ms
	POKÉ	486	972	209 ms	194 ms
	Kyber768	1184	1088	0.2 ms	0.2 ms
NIST L5	PEGASIS	≈ 512	≈ 512	∼ 120,000 ms	∼ 120,000 ms
	POKÉ	648	1296	350 ms	325 ms
	Kyber1024	1568	1568	0.3 ms	0.3 ms

Table 1: Comparison of PEGASIS, POKÉ, and Kyber (lattice) KEMs at NIST security levels 1, 3, 5.

seconds for encapsulation/decapsulation at Levels 1–5. This is already a huge improvement over earlier isogeny schemes (for instance, CSIDH at a comparable security required >1 second for key exchange), but it is still 1000× slower than Kyber. The PEGASIS prototype in Sage is even slower: on the order of tens of seconds for a single encapsulation at high security. The SageMath timings for the group action are 21 s at CSIDH-2048 (approx Level 1) and 120 s at CSIDH-4096. These are unoptimized times in a high-level language; the authors report that a prior approach (KLaPoTi) achieved 2 s at CSIDH-512 in Rust, so one can expect PEGASIS in optimized C/Rust to also speed up by one or two orders of magnitude. Even so, PEGASIS at Level 5 might remain on the order of 1 second or more per operation, which is impractical for many real-time uses.

Comparative observations: POKÉ stands out as the most efficient isogeny encryption to date, credited to its novel higher-dimensional isogeny techniques. It outperforms earlier isogeny schemes like FESTA and SQISign-based protocols by more than an order of magnitude. For example, FESTA at 128-bit security required 3 seconds to encrypt in Sage, whereas POKÉ does it in 0.1 s. QFESTA, the improved version, still had nearly double the ciphertext size and an order of magnitude slower runtime than POKÉ in tests. PEGASIS, on the other hand, is an enabling technology for group action schemes – it shows that one can in principle perform CSIDH-type key exchanges at very high security, but the raw speed is currently far behind lattice or code-based KEMs. It may be more suitable for applications where keys are exchanged less frequently or where its unique advantages (like tiny key sizes and compatibility with certain protocol constructions) outweigh the computational cost.

Kyber, representing lattice KEMs, is clearly the fastest across the board, and has moderately larger keys (a few kilobytes) which are generally acceptable in most applications. The energy and performance overhead of Kyber in TLS is minor compared to classical ECDHE. In contrast, using POKÉ or PEGASIS in TLS would currently incur a significant CPU cost (hundreds of milliseconds or more per handshake on the server side, which is likely prohibitive). While PEGASIS exhibits lower computational efficiency compared to POKÉ, it offers a significantly broader potential for extensions to advanced cryptographic functionalities enabled by group actions. This flexibility suggests that PEGASIS should be regarded as a promising candidate for integration into TLS 1.3, particularly in scenarios where functionality and long-term adaptability are prioritized alongside efficiency.

However, niche scenarios (high-security environments, memory-constrained systems where bandwidth is very limited) might still consider isogeny solutions. Furthermore, ongoing optimizations (e.g., moving POKÉ to low-level languages, exploiting parallelism in 4-dimensional isogenies) could improve their speed. The POKÉ authors note that preliminary Rust implementations support the claim that POKÉ can be made much faster outside of SageMath.

In summary, POKÉ achieves a much better performance-size tradeoff than prior isogeny schemes, coming somewhat closer to lattice KEMs in speed while beating them in size. PEGASIS achieves extremely high security in principle with small keys, but substantial further optimization is needed to make it practical. Next, we discuss how these schemes would integrate into the TLS 1.3 protocol.

5 Integration into TLS 1.3

To use PEGASIS or POKÉ in TLS 1.3, we would replace the ephemeral ECDHE key exchange with an ephemeral KEM exchange. We outline a possible integration using an approach where the client sends a KEM public key and the server responds with a ciphertext, maintaining the 1-RTT handshake:

1. **ClientHello:** The client includes a key share for an isogeny KEM (PEGASIS or POKÉ) in its ClientHello message. For example, if using POKÉ, the client can generate an ephemeral POKÉ key pair (sk_C, pk_C) (just as in the normal KEM KeyGen) and send pk_C along with an identifier of the KEM in an extension (similar to how ECDH public parameters are sent). If using PEGASIS, the client generates an ephemeral ideal $\mathfrak{b}$ and sends the corresponding public curve E_B as its key share.

2. **ServerHello:** Upon receiving the client's public key, the server performs encapsulation. For POKÉ, the server uses the client's pk_C and runs the encapsulation algorithm to produce a ciphertext C and shared secret K. It then sends C back to the client in its ServerHello (for instance, in the "key share" extension field, analogous to sending the ECDH share). For PEGASIS, the server, which has a static secret key $\mathfrak{a}$ in this scenario (or it could generate its own ephemeral if we prefer both sides ephemeral), will compute the shared secret $j(E_{AB}) = j(\mathfrak{a} \star E_B)$ and then encapsulate a key by sending some value back. In a Diffie–Hellman style KEM, the server might not need to send anything if both sides can compute the secret just from exchanging one pubkey each – however, in TLS 1.3 the server must send a "key share". In our PEGASIS KEM design, the ciphertext was E_B (the client's ephemeral), but since the client already sent that, we adapt by instead having the server send its public curve E_A (if the server uses an ephemeral secret $\mathfrak{a}$) or simply an indication that it used the client's share. A simpler approach is: let the server have a static PEGASIS pk in its certificate (though that sacrifices forward secrecy). For forward secrecy, we could do a 2-message KEM: client sends E_B, server sends E_A as its share, and both compute $j(E_{AB})$ as secret. However, that is essentially performing two-class-group operations (like a full DH exchange), not a true one-way KEM. For consistency with KEM, let's assume server static key for PEGASIS KEM in TLS handshake: the server's certificate contains E_A and in ServerHello it signals using PEGASIS; then the client encapsulates by sending E_B in ClientHello (which it already did), and server decaps. This is slightly different from normal TLS flow since the secret is known only after one message, but we can adapt the key schedule accordingly.

3. **Shared Secret Derivation:** After ServerHello, both client and server compute the shared secret. In the client-sends-pk case (e.g., POKÉ), at the moment the client receives the ServerHello with ciphertext C, it decapsulates using its ephemeral sk_C to retrieve K. The server already derived K during encapsulation. Now both have the KEM shared secret K. This K will serve as the TLS pre-handshake-secret (taking the role of the Diffie–Hellman shared secret in the HKDF-Extract that produces the handshake secret). The subsequent TLS key schedule (deriving handshake traffic keys, master secret, etc.) proceeds unchanged, except that the input secret is K instead of an ECDH Z.

4. **Transcript Hash and Finished:** All data in the handshake (the client's KEM public key and the server's ciphertext, as well as the usual messages like server certificate, etc.) are included in the running transcript hash. The finished messages are computed as normal: each side takes the handshake secret (derived from K) and the transcript hash to produce an HMAC. Because K is known to both parties and remains unknown to an attacker, the finished messages will verify correctly if and only if both sides had the same K. If an attacker modified the ciphertext C in transit, the client would decapsulate to a wrong K', causing it to compute a finished MAC that the server finds invalid, thus aborting the handshake. Similarly, if an attacker tampered with the client's public key in ClientHello, the server's derived K would not match the client's, and the client would detect a failure when it verifies the server's finished. Thus, the handshake has implicit authentication of the key exchange via the finished messages, exactly as in the ECDHE case. The use of the transcript hash (which now covers the KEM messages) ensures that any active attack on the KEM will be caught unless the attacker somehow also forges the server's finished HMAC (which is cryptographically infeasible).

5. **Certificates and Authentication:** The server will still provide a certificate and a signature over the transcript (up to ServerHello) to prove its identity, just as in standard TLS 1.3. The presence of a post-quantum KEM does not remove the need for authentication. For instance, the server might use a PQ signature like Dilithium on the handshake transcript, or during a transition, an ECDSA signature (though that would undermine quantum security if used long-term). The key point is that the KEM handles only the confidential key exchange, whereas the signature (and finished MACs) handle authentication. In some experimental designs like KEMTLS, signatures are replaced by long-term KEM keys for authentication, but that is beyond our scope; here we assume standard TLS 1.3 authentication.

Implications for TLS 1.3: The good thing is that using KEMs like PEGASIS or POKÉ in TLS 1.3 is conceptually straightforward: it can largely reuse the existing handshake flow with minimal modifications. The key schedule in TLS 1.3 is designed to be agile to different types of key exchanges – it just takes some input secret from the "KeyExchange" step and runs HKDF. Whether that secret came from ECDH or an isogeny KEM does not matter, aside from format. We would define a new TLS NamedGroup (for example, "POKÉ256" or "pegasis4096") and specify how a key share is encoded (e.g., for POKÉ, sending the curve and points in an agreed compressed format; for PEGASIS, sending the j-invariant as a 512-byte value, etc.). The client and server indicate support for these groups, and if selected, perform the above exchange.

One must consider message sizes: POKÉ's public key (sent by client) at Level 5 is 648 bytes, and ciphertext 1296 bytes. These would fit in the TLS handshake (which can carry several KB of data easily) – by comparison, sending a Kyber1024 share is 1568 bytes of public key plus 1568 bytes ciphertext if both client and server contribute, in the hybrid case possibly double that. So

bandwidth-wise, POKÉ or PEGASIS KEM in TLS is not problematic; in fact, PEGASIS keys ($256 - 512$ bytes) are even smaller, barely larger than the 32-byte ECDH keys used today. Thus, TLS packet sizes would remain low – an advantage of isogeny KEMs.

The main downside is performance: a TLS server using PEGASIS might take tens of seconds of CPU time for one handshake if not optimized, which is obviously unacceptable in practice. POKÉ is much faster, but 100 ms for decapsulation on the server side (Level 1) is still two orders of magnitude slower than X25519 ECDH (which is 0.5 ms) or Kyber (0.1 ms). A server might handle only 10 handshakes per second per core with POKÉ at that rate, vs thousands with ECDH. This could be a deployment issue except perhaps in low-connection environments or if hardware acceleration is developed.

Hybrid approaches: In an actual rollout, one might combine the KEM with classical ECDHE in a hybrid key exchange to hedge against any unforeseen weakness in the post-quantum scheme. TLS has a draft for hybrid key exchange which simply concatenates two shared secrets and feeds them into the key schedule. Using that, one could perform an ECDH and a POKÉ exchange in parallel. However, that doubles the handshake data and requires both computations, so it's more expensive. Still, given PEGASIS and POKÉ's current heavy computation, a hybrid might not add much relative cost if ECDH is negligible compared to the isogeny op.

In conclusion, integrating PEGASIS or POKÉ into TLS 1.3 is feasible by leveraging the KEM in place of ECDHE. The protocol changes are minimal: new cipher suite identifiers, and carrying the additional KEM message. The security of the handshake remains dependent on the security of the KEM and the signature scheme, with the finished message mechanism naturally extending to protect the KEM exchange. We have implicitly assumed the FO-transformed versions in this discussion, which ensures that even an active attacker tampering with the KEM messages will be detected (as decapsulation would produce the wrong key and cause handshake failure). We will now analyze the security properties of these schemes in more detail.

6 Security Analysis

6.1 Assumptions Underlying PEGASIS and POKÉ

PEGASIS's security is based on the hardness of the class group action problem: given two supersingular elliptic curves E_0 and $E_A = \mathfrak{a} \star E_0$ (where $\mathfrak{a}$ is a random ideal in the class group of an imaginary quadratic order $\mathcal{O}$), compute the ideal $\mathfrak{a}$ (or equivalently, compute an isogeny from E_0 to E_A). This is analogous to the discrete logarithm problem in Diffie–Hellman. There is no known quantum polynomial-time algorithm for this problem; the best known algorithms run in sub-exponential time [26, 12, 22] (e.g., computing the class group structure and solving a vectorization problem), but for large primes (2000+ bits) these are infeasible. Recent works suggest the problem remains hard even for quantum adversaries, with complexity estimates beyond 2^{128} for primes in the 2000–3000 bit range. PEGASIS specifically enables using very large primes by providing an efficient algorithm – it does not itself introduce a new hardness assumption, it relies on this class group action being hard (the same assumption as CSIDH). However, one must be cautious: sub-exponential algorithms (like GRH-based index calculus approaches in class groups) mean that asymptotically this problem might be weaker than ideal lattice problems. For the parameters chosen (e.g., 4096-bit p for Level 5), the authors believe it achieves the intended 256-bit quantum security margin. The indistinguishability of the shared secret $j(E_{AB})$ in the PEGASIS KEM from random is tied to the difficulty of computing any function of $\mathfrak{a}$ or $\mathfrak{b}$ given E_A, E_B (which in practice means solving the group action DDH analog).

This is formalized in some works as the "CSIDH assumption" – that $j(E_{AB})$ is pseudorandom given E_A, E_B and public parameters.

The security of POKÉ relies on a new isogeny-based hardness assumption, referred to as the *C-POKÉ problem*, which can be seen as an *Unknown Degree Isogeny Problem (UDIP)*. This problem captures the difficulty of recovering or predicting the action of an isogeny when its degree is hidden and only scaled torsion images are revealed. We recall that in SIDH-type schemes, the degree of the secret isogeny is public, whereas in POKÉ it is not, thereby strengthening the underlying assumption.

Definition 6.1 (C-POKÉ Assumption)**.** Let $E_0/\mathbb{F}_{p^2}$ be a supersingular elliptic curve with fixed torsion bases (P_0, Q_0), (R_0, S_0), (X_0, Y_0) of $E_0[\ell^e]$, for small primes $\ell \in \{2, 3, 5\}$, $e \in \{a, b, c\}$ in order respectively. Let $\phi : E_0 \to E_A$ be a secret isogeny of unknown degree $q(2^a - q)$ for some $q \in [0, 2^a]$, and let the public key include the curve E_A together with randomly scaled torsion images

$$P_A = [\alpha_2]\phi(P_0), \quad Q_A = [\beta_2]\phi(Q_0),$$
$$R_A = [\gamma_3]\phi(R_0), \quad S_A = [\gamma_3]\phi(S_0),$$
$$X_A = [\delta_5]\phi(X_0), \quad Y_A = [\delta_5]\phi(Y_0),$$

where $\alpha_2, \beta_2, \gamma_3, \delta_5 \in \mathbb{Z}_{2^a}^\times \times \mathbb{Z}_{2^a}^\times \times \mathbb{Z}_{3^b}^\times \times \mathbb{Z}_{5^e}^\times$ are random invertible scalars. Let $\psi : E_0 \to E_B$ be an isogeny of degree 3^b, and write $\psi' : E_A \to E_{AB}$ for its pushforward $\phi_*\psi$. Write

$$P_B = [\omega_2]\psi(P_0), \quad Q_B = [1/\omega_2]\psi(Q_0),$$
$$P_{AB} = [\omega_2]\psi'(P_A), \quad Q_{AB} = [1/\omega_2]\psi'(Q_A),$$
$$X_B = D_5\psi(X_0), \quad Y_B = D_5\psi(Y_0),$$
$$X_{AB} = D_5\psi'(X_A), \quad Y_{AB} = D_5\psi'(Y_A),$$

where ω_2 is a uniformly random scalar from $\mathbb{Z}_{2^a}^\times$ and D_5 is a uniformly random matrix from $\mathrm{SL}_2(\mathbb{Z}_{5^c})$. Given $(E_0, (P_0, Q_0), (R_0, S_0), (X_0, Y_0))$ and $(E_A, (P_A, Q_A), (R_A, S_A), (X_A, Y_A))$, $(E_B, (P_B, Q_B), (X_B, Y_B))$, and $(E_{AB}, (P_{AB}, Q_{AB}))$, compute the points X_{AB}, Y_{AB}.

There is currently no known attack that breaks POKÉ's assumption; the scheme was designed in response to the SIDH breaks and takes measures (unknown degree, fully revealing commutative diagram points) to avoid those exact attacks. The authors analyze its security in the random oracle model for IND-CPA (and sketch for IND-CCA with FO). As a relatively new scheme (published 2025), it has not faced as many years of cryptanalysis as, say, lattice schemes, but the early signs are positive.

6.2 IND-CPA and IND-CCA security

Both PEGASIS and POKÉ, in their base forms, provide IND-CPA security. In PEGASIS's Diffie–Hellman style exchange, an eavesdropper cannot distinguish the shared $j(E_{AB})$ from random without solving the underlying group action problem. POKÉ, being an encryption scheme, was explicitly shown to be IND-CPA under its new assumption (roughly, a reduction exists from an isogeny problem to breaking POKÉ encryption).

When we apply the Fujisaki–Okamoto transform, we aim for IND-CCA2 security. FO has been proven to work for many schemes in the ROM – assuming the base encryption is IND-CPA and has certain smoothness in its decryption behavior (no decryption oracles other than by trial), the FO-transformed KEM is IND-CCA in the ROM. For POKÉ-KEM, we would

rely on the fact that POKÉ's encryption is sufficiently randomness-dependent and checkable such that a decrypted plaintext can be re-encrypted and verified. By including the hash of the plaintext (or some checksum in the ciphertext), as we did with H_m in the pseudocode, decapsulation will detect any tampering with all but negligible probability, thus preventing an attacker from querying the decapsulation oracle in a way that yields useful information. Therefore, POKÉ-FO-KEM should be IND-CCA secure in ROM, under the assumption that hashing is a random oracle and the POKÉ problem is hard.

For PEGASIS-KEM, FO transform also should confer IND-CCA. The one caveat is that PEGASIS's ciphertext is just an elliptic curve point E_B. Without FO, an attacker could send a random curve E'_B to a decapsulation oracle; the decapsulator (with secret a) would compute $j(a \star E'_B)$ and return the hash. This potentially could act as an oracle solving some problem (though likely not, since $a \star E'_B$ is essentially a random curve if E'_B is random, yielding a random hash that doesn't give the attacker any edge). Nonetheless, FO shuts down any structure by making the output key independent of the raw decapsulation if the ciphertext is invalid. In ROM, we assume the hash is random, so a malformed E'_B yields a random K' that's independent of the secret. Thus IND-CCA holds. In QROM, security is more subtle: FO's proof in the quantum setting often requires either a slight tweak (Fujisaki–Okamoto transform has variants proven secure in QROM, such as using a double hash or specific random oracle programming). The standard FO as applied should still be secure in QROM for these schemes, but a formal proof would require careful analysis (some works by Targhi–Unruh [39] and Jiang et al. [23] have looked at FO in QROM and found it secure under slightly stronger assumptions or minor modifications).

We note that the designs of both PEGASIS and POKÉ avoid any obvious structured plaintext that an attacker could exploit in a CCA scenario. For example, there is no "decryption failure" probability or unnatural error patterns (as sometimes occur in lattice schemes) – decryption/decapsulation is always deterministic given the correct inputs. This simplifies the CCA analysis: the only way an attacker can try to differentiate keys is by crafting a ciphertext that yields a predictable decapsulation difference. FO transform ensures that any such attempt either yields a random key (if they deviate from a valid ciphertext) or yields exactly the same key the legitimate encapsulation would (if they somehow guess a valid ciphertext).

6.3 ROM vs QROM

The Random Oracle Model is used in both schemes: hash functions in FO, and also in some internal compressions in POKÉ, are modeled as random oracles for security proofs. This is common in post-quantum KEMs (Kyber included). The concern is that a quantum adversary could query these oracles in superposition, potentially breaking some security arguments. Security in the Quantum Random Oracle Model (QROM) is typically weaker to prove. However, there has been considerable research showing that many ROM proofs can be translated to QROM with some loss of tightness or additional assumptions. For FO specifically, the original FO transform's proof was not directly QROM-secure, but a subsequent work by Jiang et al. [23] showed FO (with an appropriate usage) is secure in QROM under IND-CPA and one-wayness assumptions of the base scheme, using the "quantum-aware" notion of one-wayness. Essentially, as long as the scheme's ciphertext has sufficient entropy (so that hashing the plaintext/randomness binds the output), FO can be shown secure in QROM. We expect that POKÉ and PEGASIS meet those criteria (the outputs are large elliptic curve points, etc.). There are no known quantum attacks that exploit the random oracle in these schemes beyond the generic Grover's algorithm (which would at most quadratically weaken the hash, easily mitigated by using larger hash

output, e.g., 256-bit to maintain 128-bit security under Grover).

In summary, we consider PEGASIS-FO-KEM and POKÉ-FO-KEM to be IND-CCA2 secure under their respective hard problems in the ROM, and we have confidence (though not a full formal proof in this text) that this security extends to the QROM. The use of transcript hashing in TLS adds an additional safety net: even if an attacker could somehow find a subtle way to get information from decapsulation (which FO should prevent), the handshake's finished MAC would catch any tampering. Thus, when integrated into TLS, the primary requirement is that the KEM is IND-CCA to protect the handshake secret – which is achieved via FO.

6.4 Additional Security Considerations

1. **Forward Secrecy:** Both PEGASIS and POKÉ, when used ephemerally in TLS, could provide forward secrecy. If long-term keys are compromised after the fact, past session keys remain safe (assuming the ephemeral secrets b or client's POKÉ key are erased). If the server uses a static PEGASIS key (less ideal), it would lose forward secrecy for that scheme (like a static DH would). We prefer ephemeral usage for forward secrecy.

2. **Side Channels:** An important practical aspect: isogeny computations involve complex arithmetic that could be vulnerable to timing or cache attacks. Implementations must use constant-time operations for secret isogeny steps. This is similar to implementing ECDH or lattice ops, but the complexity of isogeny formulas (especially 4-dim theta coordinates in POKÉ, etc.) is non-trivial. Research on side-channel defenses for isogeny crypto is ongoing, but not as mature as for ECDH or lattice (which have had more years of study). This paper does not delve into implementation security, but it's worth noting that any adoption of PEGASIS/POKÉ would require careful analysis of side-channel leakage.

3. **Global compatibility:** Using these schemes in TLS doesn't introduce new trust assumptions beyond what TLS already has (trusted CA for certs, etc.). It's purely a performance and security trade-off: smaller handshake messages vs heavier computations.

In conclusion, both PEGASIS and POKÉ can be transformed into KEMs that meet strong security definitions (IND-CCA) and can be safely used in protocols like TLS 1.3. They rely on well-defined hard problems believed to be quantum-resistant. POKÉ's assumption is newer but appears solid and has undergone initial peer review (Eurocrypt 2025). PEGASIS builds on the established CSIDH problem; its main novelty is making high-security instances feasible, not altering the problem's nature.

7 Conclusion

We have presented an in-depth examination of PEGASIS and POKÉ, two state-of-the-art isogeny-based cryptosystems, covering their implementation details, KEM transformations, performance characteristics, and integration into TLS 1.3. PEGASIS demonstrates that effective class group actions using 4-dimensional isogenies are practical up to very high security ($\approx$ CSIDH-4096), offering compact keys but with significant computational cost. POKÉ introduces a hybrid SIDH/higher-dimensional approach that achieves unprecedented efficiency for an isogeny scheme, making isogeny encryption competitive in size and even performance-wise narrowing the gap to lattices.

Our comparison shows that at NIST Level 1, POKÉ's ciphertext is about 15% smaller than Kyber's, and its public key less than half the size of Kyber's, though encapsulation is about

10^3 times slower. PEGASIS (with large CSIDH parameters) pushes key sizes down further (a few hundred bytes) but currently at a heavy speed trade-off (orders of magnitude slower). In a TLS 1.3 setting, replacing ECDHE with POKÉ or PEGASIS KEM is straightforward from a protocol perspective, and the handshake retains its security structure (with the KEM providing confidentiality and the TLS transcript providing authenticity). The primary challenges are performance and implementation complexity.

Both schemes attain IND-CCA security through the Fujisaki–Okamoto transform, and the security relies on problems that are conjectured to resist quantum attacks. We discussed how these hold up in the ROM and QROM, and conclude that no fundamental weaknesses are evident – though as with any new cryptosystem, especially one with novel assumptions (POKÉ), continued cryptanalysis is essential.

On the theoretical side, a formal proof of FO-CCA security in the QROM for POKÉ-KEM and PEGASIS-KEM would strengthen confidence. On the implementation side, optimizing these schemes in low-level languages (C, assembly) and perhaps leveraging parallelism (e.g., multi-core or vector instructions for the 4-dimensional isogeny steps) could drastically reduce the encapsulation/decapsulation times. If POKÉ's operations can be brought to just a few milliseconds, it might become a viable alternative in certain applications where bandwidth is at a premium. Additionally, exploring hybrid schemes (combining lattice and isogeny secrets) could yield practical transitional mechanisms – one could take advantage of the small keys of POKÉ while leaning on the speed of lattices, getting the best of both in a TLS handshake.

Finally, standardization and interoperability testing (e.g., implementing POKÉ in TLS libraries, measuring handshake latency) would be necessary steps should these schemes be considered for real-world deployment. As NIST continues a fourth round for alternate KEMs, schemes like POKÉ (or its successors) might enter the fray, offering diversity against the predominant lattice-based approaches. PEGASIS, while perhaps too slow for mainstream use, could find niche use in ultra-high security contexts or serve as a foundation for other primitives (like isogeny-based signatures or group protocols) where its effective group action is crucial.

In summary, PEGASIS and POKÉ expand the toolkit of post-quantum cryptography, illustrating the ongoing innovation in isogeny-based methods. They show that even after the setback of SIDH's cryptanalysis, the isogeny field is far from exhausted – with new ideas enabling both higher security and higher performance. The integration and analysis we provided here aims to aid in understanding how such schemes can be employed in practice, and what trade-offs they entail. The pursuit of quantum-safe and efficient key exchange continues, and these two schemes represent important milestones in that journey.

Acknowledgment

This work was in part supported by JSPS KAKENHI Grant Number JP23K24846. This work was in part supported by JST K Program Grant Number JPMJKP24U2, Japan. The authors would like to express sincere gratitude to Shingo Sato for his valuable advice and insightful suggestions throughout this work.

References

[1] Marius A Aardal, Andrea Basso, Luca De Feo, Sikhar Patranabis, and Benjamin Wesolowski. A complete security proof of sqisign. *Cryptology ePrint Archive*, 2025.

[2] Michel Abdalla, Thorsten Eisenhofer, Eike Kiltz, Sabrina Kunzweiler, and Doreen Riepel. Password-authenticated key exchange from group actions. In *Annual international cryptology conference*, pages 699–728. Springer, 2022.

[3] Bill Allombert, Jean-François Biasse, Jonathan Komada Eriksen, Péter Kutas, Chris Leonardi, Aurel Page, Renate Scheidler, and Márton Tot Bagi. PEARL-SCALLOP: Parameter extension applicable in real-life SCALLOP. Cryptology ePrint Archive, Paper 2024/1744, 2024.

[4] Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. Csi-shark: Csi-fish with sharing-friendly keys. In *Australasian Conference on Information Security and Privacy*, pages 471–502. Springer, 2023.

[5] Andrea Basso and Luciano Maino. Poké: A compact and efficient pke from higher-dimensional isogenies. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 94–123. Springer, 2025.

[6] Andrea Basso, Luciano Maino, and Giacomo Pope. FESTA: fast encryption from supersingular torsion attacks. In *ASIACRYPT (7)*, volume 14444 of *LNCS*, pages 98–126. Springer, 2023.

[7] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. Csi-fish: Efficient isogeny based signatures through class group computations. In *ASIACRYPT (1)*, volume 11921 of *LNCS*, pages 227–247. Springer, 2019.

[8] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In *EUROCRYPT (5)*, volume 14008 of *LNCS*, pages 423–447. Springer, 2023.

[9] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In *ASIACRYPT (3)*, volume 11274 of *LNCS*, pages 395–427. Springer, 2018.

[10] Denis X Charles, Kristin E Lauter, and Eyal Z Goren. Cryptographic hash functions from expander graphs. *Journal of CRYPTOLOGY*, 22(1):93–113, 2009.

[11] Mingjie Chen, Antonin Leroux, and Lorenz Panny. Scallop-hd: group action from 2-dimensional isogenies. In *IACR International Conference on Public-Key Cryptography*, pages 190–216. Springer, 2024.

[12] Andrew Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *Journal of Mathematical Cryptology*, 8(1):1–29, 2014.

[13] Jean Marc Couveignes. Hard homogeneous spaces. *IACR Cryptol. ePrint Arch.*, page 291, 2006.

[14] Pierrick Dartois, Jonathan Komada Eriksen, Tako Boris Fouotsa, Arthur Herlédan Le Merdy, Riccardo Invernizzi, Damien Robert, Ryan Rueger, Frederik Vercauteren, and Benjamin Wesolowski. Pegasis: Practical effective class group action using 4-dimensional isogenies. In *Annual International Cryptology Conference*, pages 67–99. Springer, 2025.

[15] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the tls 1.3 handshake protocol. *Journal of Cryptology*, 34(4):37, 2021.

[16] Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski. SCALLOP: scaling the csi-fish. In *Public Key Cryptography (1)*, volume 13940 of *LNCS*, pages 345–375. Springer, 2023.

[17] Luca De Feo and Steven D. Galbraith. Seasign: Compact isogeny signatures from class group actions. *IACR Cryptol. ePrint Arch.*, page 824, 2018.

[18] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *J. Math. Cryptol.*, 8(3):209–247, 2014.

[19] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. Sqisign: Compact post-quantum signatures from quaternions and isogenies. In *ASIACRYPT (1)*, volume 12491 of *LNCS*, pages 64–93. Springer, 2020.

[20] Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In *Public Key Cryptography (2)*, volume 12111 of *LNCS*, pages 187–212. Springer, 2020.

[21] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic

curve isogenies. In *PQCrypto*, volume 7071 of *LNCS*, pages 19–34. Springer, 2011.

[22] David Jao, Jason LeGrow, Christopher Leonardi, and Luis Ruiz-Lopez. A subexponential-time, polynomial quantum space algorithm for inverting the cm group action. *Journal of Mathematical Cryptology*, 14(1):129–138, 2020.

[23] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. Ind-cca-secure key encapsulation mechanism in the quantum random oracle model, revisited. In *CRYPTO (3)*, volume 10993 of *LNCS*, pages 96–125. Springer, 2018.

[24] Ernst Kani. The number of curves of genus two with elliptic differentials. 1997.

[25] Shuichi Katsumata, Yi-Fu Lai, Jason T LeGrow, and Ling Qin. Csi-otter: Isogeny-based (partially) blind signatures from the class group action with a twist. *Designs, Codes and Cryptography*, 92(11):3587–3643, 2024.

[26] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005.

[27] Antonin Leroux and Maxime Roméas. Updatable encryption from group actions. In *International Conference on Post-Quantum Cryptography*, pages 20–53. Springer, 2024.

[28] Luciano Maino and Chloe Martindale. An attack on SIDH with arbitrary starting curve. *IACR Cryptol. ePrint Arch.*, page 1026, 2022.

[29] Tomoki Moriya. Lit-sigamal: An efficient isogeny-based pke based on a lit diagram. *Cryptology ePrint Archive*, 2024.

[30] Tomoki Moriya, Hiroshi Onuki, and Tsuyoshi Takagi. Sigamal: A supersingular isogeny-based PKE and its application to a PRF. In *ASIACRYPT (2)*, volume 12492 of *LNCS*, pages 551–580. Springer, 2020.

[31] Kohei Nakagawa and Hiroshi Onuki. Qfesta: Efficient algorithms and parameters for festa using quaternion algebras. In *Annual International Cryptology Conference*, pages 75–106. Springer, 2024.

[32] NIST. Post-quantum cryptography: Additional digital signature schemes. https://csrc.nist.gov/projects/pqc-dig-sig, 2023.

[33] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. fips203, 2023.

[34] Aurel Page and Damien Robert. Introducing clapoti (s): Evaluating the isogeny class group action in polynomial time. *Cryptology ePrint Archive*, 2023.

[35] Lorenz Panny, Christophe Petit, and Miha Stopar. Klapoti: an asymptotically efficient isogeny group action from 2-dimensional isogenies. *Cryptology ePrint Archive*, 2024.

[36] Damien Robert. Breaking SIDH in polynomial time. In *EUROCRYPT (5)*, volume 14008 of *LNCS*, pages 472–503. Springer, 2023.

[37] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. *IACR Cryptol. ePrint Arch.*, page 145, 2006.

[38] Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer, 2009.

[39] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the fujisaki-okamoto and oaep transforms. In *Theory of Cryptography Conference*, pages 192–216. Springer, 2016.

[40] Yunxiao Zhou, Shengli Liu, and Shuai Han. Multi-hop fine-grained proxy re-encryption. In *Public Key Cryptography (4)*, volume 14604 of *LNCS*, pages 161–192. Springer, 2024.